OpenSMT and Applications to Interpolation and Proof Manipulation

Roberto Bruttomesso, Natasha Sharygina

USI Lugano

MIT - June 16, 2011

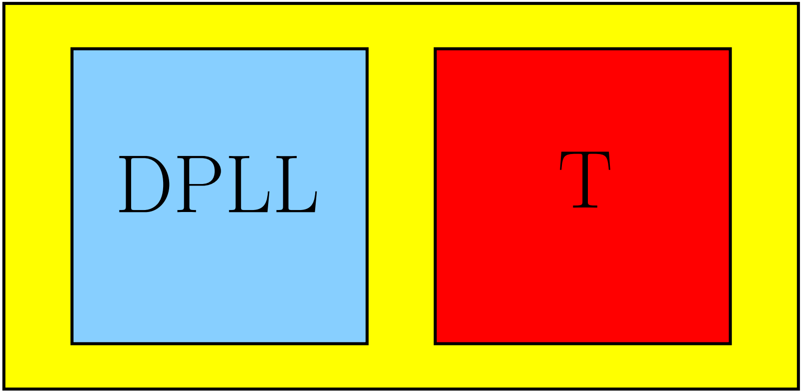
Outline

1. The OpenSMT Solver
2. Interpolants
3. Application to Program Verification
4. Computing Interpolants
5. Proof Transformation (for interpolation and reduction)

The OpenSMT Solver

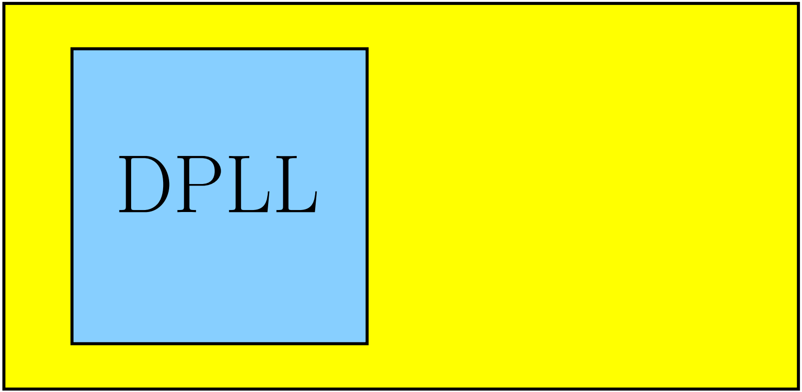
Introduction

$$e(\mathrm{DPLL}(T)) = e(\mathrm{DPLL}) + e(T) + e(\mathrm{COMM})$$



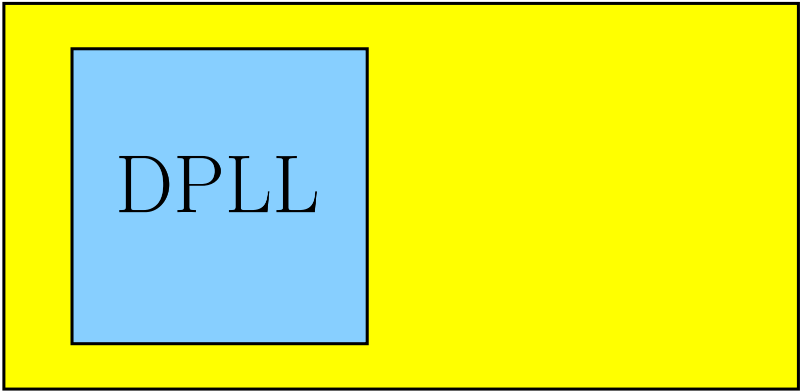
Introduction

$$e(\text{DPLL}(T)) = \boxed{e(\text{DPLL})} + \boxed{e(T)} + \boxed{e(\text{COMM})}$$



Introduction

$$e(\mathrm{DPLL}(\mathrm{T})) \approx \boxed{e(\mathrm{T})}$$



Join SMT-COMP !

The OpenSMT Solver

- Open-source solver developed at USI since 2008[1]
- Based on MiniSAT, and Efficient (e.g., see SMT-COMP'10)

[1] Available at `http://www.verify.usi.ch/opensmt`

The OpenSMT Solver

- Open-source solver developed at USI since 2008[1]
- Based on MiniSAT, and Efficient (e.g., see SMT-COMP'10)
- Structured to be easily extended with new theory-solvers

[1] Available at `http://www.verify.usi.ch/opensmt`

The OpenSMT Solver

- Open-source solver developed at USI since 2008[1]
- Based on MiniSAT, and Efficient (e.g., see SMT-COMP'10)
- Structured to be easily extended with new theory-solvers
- Several algorithms for computing interpolants and manipulating proofs of unsatisfiability

[1] Available at `http://www.verify.usi.ch/opensmt`

The OpenSMT Solver

- Open-source solver developed at USI since 2008[1]
- Based on MiniSAT, and Efficient (e.g., see SMT-COMP'10)
- Structured to be easily extended with new theory-solvers
- Several algorithms for computing interpolants and manipulating proofs of unsatisfiability
- Coming soon: integration with model-checker MCMT (JWW F.Alberti, S. Ghilardi, S.Ranise)

[1] Available at `http://www.verify.usi.ch/opensmt`

Interpolants

Quantifier-free Interpolation

A first-order theory T has **quantifier-free interpolation** property

Quantifier-free Interpolation

A first-order theory T has **quantifier-free interpolation** property

iff

for every quantifier-free formulae A, B, such that $A \wedge B$ is T-unsatisfiable, there exists a quantifier-free formula I such that:

Quantifier-free Interpolation

A first-order theory T has **quantifier-free interpolation** property

iff

for every quantifier-free formulae A, B, such that $A \wedge B$ is T-unsatisfiable, there exists a quantifier-free formula I such that:

(i) $T \vdash A \rightarrow I$;

(ii) $B \wedge I$ is T-unsatisfiable;

(iii) I is defined over common symbols of A and B.

Quantifier-free Interpolation

A first-order theory T has **quantifier-free interpolation** property

iff

for every quantifier-free formulae A, B, such that $A \wedge B$ is T-unsatisfiable, there exists a quantifier-free formula I such that:

(i) $T \vdash A \rightarrow I$;

(ii) $B \wedge I$ is T-unsatisfiable;

(iii) I is defined over common symbols of A and B.

In short, I is an **overapproximation** of A that is still unsatisfiable with B, and that uses the common language

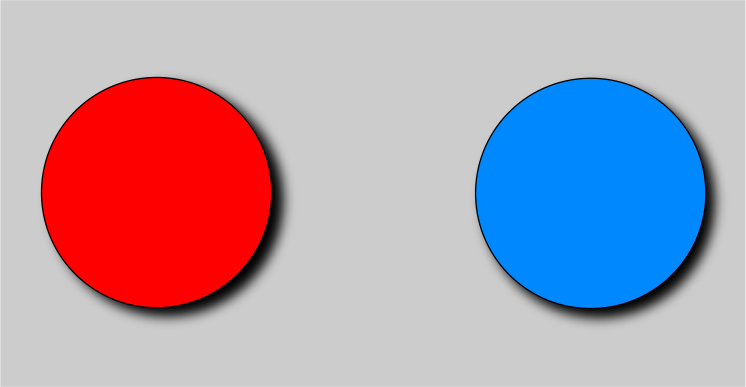
Quantifier-free Interpolation

For $A \wedge B$ is T-unsatisfiable, I is a quantifier-free formula such that:

(i) $T \vdash A \to I$;

(ii) $B \wedge I$ is T-unsatisfiable;

(iii) I is defined over common symbols of A and B.



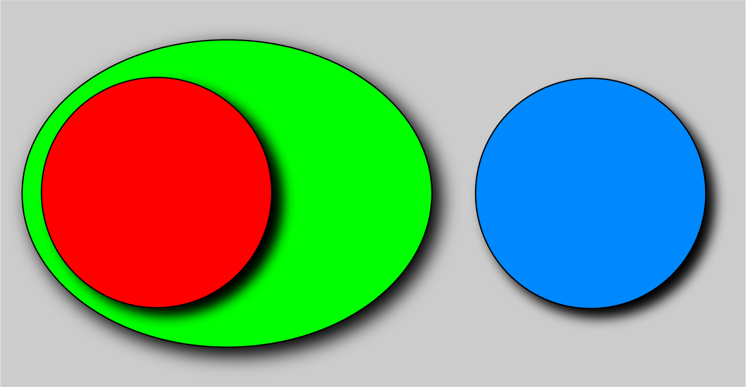
Quantifier-free Interpolation

For $A \wedge B$ is T-unsatisfiable, I is a quantifier-free formula such that:

(i) $T \vdash A \rightarrow I$;

(ii) $B \wedge I$ is T-unsatisfiable;

(iii) I is defined over common symbols of A and B.



Theories with Quantifier-free Interpolation

Many useful theories used in software verification admit quantifier-free interpolants:

Theories with Quantifier-free Interpolation

Many useful theories used in software verification admit quantifier-free interpolants:

- Linear Real Arithmetic ($\mathcal{LRA}$);

Theories with Quantifier-free Interpolation

Many useful theories used in software verification admit quantifier-free interpolants:

- Linear Real Arithmetic ($\mathcal{LRA}$);
- Linear Integer Arithmetic ($\mathcal{LIA}$)

Theories with Quantifier-free Interpolation

Many useful theories used in software verification admit quantifier-free interpolants:

- Linear Real Arithmetic ($\mathcal{LRA}$);
- Linear Integer Arithmetic ($\mathcal{LIA}$)
  (with help of $\{\equiv_n\}$ predicates);

Theories with Quantifier-free Interpolation

Many useful theories used in software verification admit quantifier-free interpolants:

- Linear Real Arithmetic ($\mathcal{LRA}$);
- Linear Integer Arithmetic ($\mathcal{LIA}$)
  (with help of $\{\equiv_n\}$ predicates);
- Equality with Uninterpreted Functions ($\mathcal{EUF}$);

Theories with Quantifier-free Interpolation

Many useful theories used in software verification admit quantifier-free interpolants:

- Linear Real Arithmetic ($\mathcal{LRA}$);
- Linear Integer Arithmetic ($\mathcal{LIA}$)
  (with help of $\{\equiv_n\}$ predicates);
- Equality with Uninterpreted Functions ($\mathcal{EUF}$);
- Arrays with extensionality ($\mathcal{AX}$)

Theories with Quantifier-free Interpolation

Many useful theories used in software verification admit quantifier-free interpolants:

- Linear Real Arithmetic ($\mathcal{LRA}$);
- Linear Integer Arithmetic ($\mathcal{LIA}$)
  (with help of $\{\equiv_n\}$ predicates);
- Equality with Uninterpreted Functions ($\mathcal{EUF}$);
- Arrays with extensionality ($\mathcal{AX}$)
  (with help of `diff` function);

Theories with Quantifier-free Interpolation

Many useful theories used in software verification admit quantifier-free interpolants:

- Linear Real Arithmetic ($\mathcal{LRA}$);
- Linear Integer Arithmetic ($\mathcal{LIA}$)
  (with help of $\{\equiv_n\}$ predicates);
- Equality with Uninterpreted Functions ($\mathcal{EUF}$);
- Arrays with extensionality ($\mathcal{AX}$)
  (with help of `diff` function);
- some combinations, like ($\mathcal{LRA} \cup \mathcal{EUF}$);

Theories with Quantifier-free Interpolation

Many useful theories used in software verification admit quantifier-free interpolants:

- Linear Real Arithmetic ($\mathcal{LRA}$);
- Linear Integer Arithmetic ($\mathcal{LIA}$)
  (with help of $\{\equiv_n\}$ predicates);
- Equality with Uninterpreted Functions ($\mathcal{EUF}$);
- Arrays with extensionality ($\mathcal{AX}$)
  (with help of `diff` function);
- some combinations, like ($\mathcal{LRA} \cup \mathcal{EUF}$);
- but not some other, like ($\mathcal{LIA} \cup \mathcal{EUF}$).

Theories with Quantifier-free Interpolation

Many useful theories used in software verification admit quantifier-free interpolants:

- Linear Real Arithmetic ($\mathcal{LRA}$);
- Linear Integer Arithmetic ($\mathcal{LIA}$)
  (with help of $\{\equiv_n\}$ predicates);
- Equality with Uninterpreted Functions ($\mathcal{EUF}$);
- Arrays with extensionality ($\mathcal{AX}$)
  (with help of `diff` function);
- some combinations, like ($\mathcal{LRA} \cup \mathcal{EUF}$);
- but not some other, like ($\mathcal{LIA} \cup \mathcal{EUF}$).

In general, those theories that admit **Quantifier Elimination**, also admit quantifier-free interpolants

Some easy examples

Example (Trivial cases)

If A is unsatisfiable on its own (i.e., $A = \bot$), then $I = \bot$.

Some easy examples

Example (Trivial cases)

If A is unsatisfiable on its own (i.e., $A = \bot$), then $I = \bot$.
If B is unsatisfiable on its own (i.e., $B = \bot$), then $I = \top$.

Some easy examples

Example (Trivial cases)

If A is unsatisfiable on its own (i.e., $A = \bot$), then $I = \bot$.
If B is unsatisfiable on its own (i.e., $B = \bot$), then $I = \top$.

Example (Boolean logic)

$A \equiv \{\neg a \wedge (a \vee c_1) \wedge (a \vee c_2)\}$
$B \equiv \{\neg b \wedge (b \vee \neg c_1) \wedge (b \vee \neg c_2)\}$

Some easy examples

Example (Trivial cases)

If A is unsatisfiable on its own (i.e., $A = \bot$), then $I = \bot$.
If B is unsatisfiable on its own (i.e., $B = \bot$), then $I = \top$.

Example (Boolean logic)

$A \equiv \{\neg a \wedge (a \vee c_1) \wedge (a \vee c_2)\}$
$B \equiv \{\neg b \wedge (b \vee \neg c_1) \wedge (b \vee \neg c_2)\}$
$I = \{c_1\}$

Some easy examples

Example (Trivial cases)

If A is unsatisfiable on its own (i.e., $A = \bot$), then $I = \bot$.
If B is unsatisfiable on its own (i.e., $B = \bot$), then $I = \top$.

Example (Boolean logic)

$A \equiv \{\neg a \wedge (a \vee c_1) \wedge (a \vee c_2)\}$
$B \equiv \{\neg b \wedge (b \vee \neg c_1) \wedge (b \vee \neg c_2)\}$
$I = \{c_1\}$ (one of the many possible, see later)

Some easy examples

Example (Trivial cases)

If A is unsatisfiable on its own (i.e., $A = \bot$), then $I = \bot$.
If B is unsatisfiable on its own (i.e., $B = \bot$), then $I = \top$.

Example (Boolean logic)

$A \equiv \{\neg a \wedge (a \vee c_1) \wedge (a \vee c_2)\}$
$B \equiv \{\neg b \wedge (b \vee \neg c_1) \wedge (b \vee \neg c_2)\}$
$I = \{c_1\}$ (one of the many possible, see later)

Example (Linear Real Arithmetic)

$A \equiv \{(x - y \leq 2) \wedge (y - z \leq 1)\}$
$B \equiv \{(z - w \leq 0) \wedge (w - x \leq -10)\}$

Some easy examples

Example (Trivial cases)

If A is unsatisfiable on its own (i.e., $A = \bot$), then $I = \bot$.
If B is unsatisfiable on its own (i.e., $B = \bot$), then $I = \top$.

Example (Boolean logic)

$A \equiv \{\neg a \wedge (a \vee c_1) \wedge (a \vee c_2)\}$
$B \equiv \{\neg b \wedge (b \vee \neg c_1) \wedge (b \vee \neg c_2)\}$
$I = \{c_1\}$ (one of the many possible, see later)

Example (Linear Real Arithmetic)

$A \equiv \{(x - y \leq 2) \wedge (y - z \leq 1)\}$
$B \equiv \{(z - w \leq 0) \wedge (w - x \leq -10)\}$
$I = \{x - z \leq 8\}$

Some easy examples

Example (Trivial cases)

If A is unsatisfiable on its own (i.e., $A = \bot$), then $I = \bot$.
If B is unsatisfiable on its own (i.e., $B = \bot$), then $I = \top$.

Example (Boolean logic)

$A \equiv \{\neg a \wedge (a \vee c_1) \wedge (a \vee c_2)\}$
$B \equiv \{\neg b \wedge (b \vee \neg c_1) \wedge (b \vee \neg c_2)\}$
$I = \{c_1\}$ (one of the many possible, see later)

Example (Linear Real Arithmetic)

$A \equiv \{(x - y \leq 2) \wedge (y - z \leq 1)\}$
$B \equiv \{(z - w \leq 0) \wedge (w - x \leq -10)\}$
$I = \{x - z \leq 8\}$ (one of the infinite possible)

Computing interpolants with OpenSMT

SMT-LIB 2 Standard does not support (yet) interpolation commands

Computing interpolants with OpenSMT

SMT-LIB 2 Standard does not support (yet) interpolation commands

OpenSMT supports **non-standard** interpolation commands

Computing interpolants with OpenSMT

SMT-LIB 2 Standard does not support (yet) interpolation commands

OpenSMT supports **non-standard** interpolation commands

- `(set-option :produce-interpolants <bool>)`
  tells OpenSMT to compute interpolants

Computing interpolants with OpenSMT

SMT-LIB 2 Standard does not support (yet) interpolation commands

OpenSMT supports **non-standard** interpolation commands

- `(set-option :produce-interpolants <bool>)`
  tells OpenSMT to compute interpolants

- `(assert-partition <formula>)`
  tells OpenSMT about a partition

Computing interpolants with OpenSMT

SMT-LIB 2 Standard does not support (yet) interpolation commands

OpenSMT supports **non-standard** interpolation commands

- `(set-option :produce-interpolants <bool>)`
  tells OpenSMT to compute interpolants
- `(assert-partition <formula>)`
  tells OpenSMT about a partition
- `(get-interpolant <n>)`
  command to retrieve an interpolant

Demo Time

Application to Program Verification

Application to Program Verification

So far we have considered interpolants between two partitions A and B

A more **general definition** involves $n \geq 2$ partitions $A_1, \ldots, A_n$, whose conjunction is unsatisfiable

Application to Program Verification

So far we have considered interpolants between two partitions A and B

A more **general definition** involves $n \geq 2$ partitions $A_1, \ldots, A_n$, whose conjunction is unsatisfiable

Interpolants $I_0, \ldots, I_n$ are such that

(i) $I_0 = \top$, $I_n = \bot$;

(ii) $T \vdash (I_k \land A_{k+1}) \to I_{k+1}$;

(iii) I_k on shared symbols of A_k and A_{k+1}.

For $n = 2$, you get the previous definition for A and B

Application to Program Verification

Lazy Abstraction with Interpolants

Original (Concrete) Program

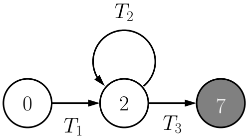
```
y = x;
while ( x ≥ 1 ) {
      x = x - 1;
      y = y - 1;
}
if ( y ≥ 1 )
      ERROR;
```

Application to Program Verification

Lazy Abstraction with Interpolants

Original (Concrete) Program	Control Flow and Transitions

```
1:  y = x;
2:  while ( x ≥ 1 ) {
3:      x = x - 1;
4:      y = y - 1;
5:  }
6:  if ( y ≥ 1 )
7:      ERROR;
```



$$T_1\colon\ \top \wedge \begin{cases} x' := x \\ y' := x \end{cases}$$

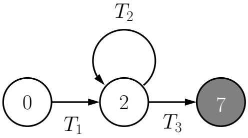
$$T_2\colon\ x \geq 1 \wedge \begin{cases} x' := x - 1 \\ y' := y - 1 \end{cases}$$

$$T_3\colon\ x \leq 0 \wedge y \geq 1 \wedge \begin{cases} x' := x \\ y' := y \end{cases}$$

Application to Program Verification

Lazy Abstraction with Interpolants

(Abstract) Program Unwinding	Control Flow and Transitions



T_1: $\top \wedge \begin{cases} x' := x \\ y' := x \end{cases}$

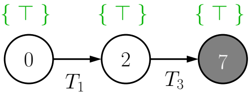
T_2: $x \geq 1 \wedge \begin{cases} x' := x - 1 \\ y' := y - 1 \end{cases}$

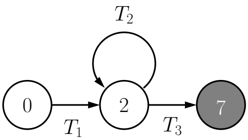
T_3: $x \leq 0 \wedge y \geq 1 \wedge \begin{cases} x' := x \\ y' := y \end{cases}$

Application to Program Verification

Lazy Abstraction with Interpolants

(Abstract) Program Unwinding	Control Flow and Transitions



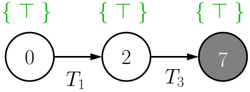


$$T_1\colon\ \top \wedge \begin{cases} x' := x \\ y' := x \end{cases}$$

$$T_2\colon\ x \geq 1 \wedge \begin{cases} x' := x - 1 \\ y' := y - 1 \end{cases}$$

$$T_3\colon\ x \leq 0 \wedge y \geq 1 \wedge \begin{cases} x' := x \\ y' := y \end{cases}$$

(Abstract) Program Unwinding	Control Flow and Transitions



true

$\mathrm{x}_1 = \mathrm{x}_0$

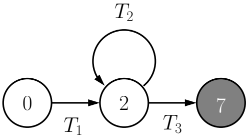
$\mathrm{y}_1 = \mathrm{x}_0$

$\mathrm{x}_1 \leq 0$

$\mathrm{y}_1 \geq 1$

$\mathrm{x}_2 = \mathrm{x}_1$

$\mathrm{y}_2 = \mathrm{y}_1$

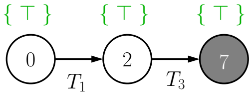


$$T_1\colon\ \top \wedge \begin{cases} x' := x \\ y' := x \end{cases}$$

$$T_2\colon\ x \geq 1 \wedge \begin{cases} x' := x - 1 \\ y' := y - 1 \end{cases}$$

$$T_3\colon\ x \leq 0 \wedge y \geq 1 \wedge \begin{cases} x' := x \\ y' := y \end{cases}$$

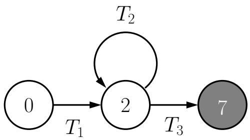
(Abstract) Program Unwinding	Control Flow and Transitions



$\{\ \top\ \}$

true
$x_1 = x_0$
$y_1 = x_0$

$x_1 \leq 0$
$y_1 \geq 1$
$x_2 = x_1$
$y_2 = y_1$

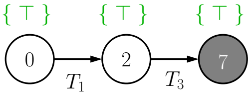


$$T_1\colon\ \top \wedge \begin{cases} x' := x \\ y' := x \end{cases}$$

$$T_2\colon\ x \geq 1 \wedge \begin{cases} x' := x - 1 \\ y' := y - 1 \end{cases}$$

$$T_3\colon\ x \leq 0 \wedge y \geq 1 \wedge \begin{cases} x' := x \\ y' := y \end{cases}$$

(Abstract) Program Unwinding



{ ⊤ }

true

$x_1 = x_0$

$y_1 = x_0$

{ y_1 - $x_1 \leq 0$ }

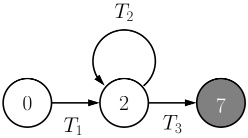
$x_1 \leq 0$

$y_1 \geq 1$

$x_2 = x_1$

$y_2 = y_1$

Control Flow and Transitions



$$T_1\colon\ \top \wedge \begin{cases} x' := x \\ y' := x \end{cases}$$

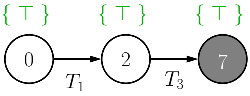
$$T_2\colon\ x \geq 1 \wedge \begin{cases} x' := x - 1 \\ y' := y - 1 \end{cases}$$

$$T_3\colon\ x \leq 0 \wedge y \geq 1 \wedge \begin{cases} x' := x \\ y' := y \end{cases}$$

Application to Program Verification

Lazy Abstraction with Interpolants

(Abstract) Program Unwinding



{ ⊤ }

true

$x_1 = x_0$

$y_1 = x_0$

{ y_1 - $x_1 \leq 0$ }

$x_1 \leq 0$

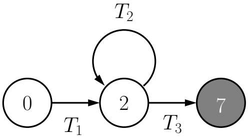
$y_1 \geq 1$

$x_2 = x_1$

$y_2 = y_1$

{ ⊥ }

Control Flow and Transitions



$$T_1\colon\ \top \wedge \begin{cases} x' := x \\ y' := x \end{cases}$$

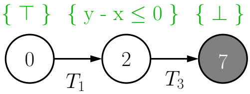
$$T_2\colon\ x \geq 1 \wedge \begin{cases} x' := x - 1 \\ y' := y - 1 \end{cases}$$

$$T_3\colon\ x \leq 0 \wedge y \geq 1 \wedge \begin{cases} x' := x \\ y' := y \end{cases}$$

Application to Program Verification

Lazy Abstraction with Interpolants

(Abstract) Program Unwinding	Control Flow and Transitions



$\{ \top \}$

true

$x_1 = x_0$

$y_1 = x_0$

$\{ y_1 - x_1 \leq 0 \}$

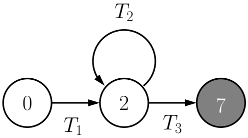
$x_1 \leq 0$

$y_1 \geq 1$

$x_2 = x_1$

$y_2 = y_1$

$\{ \bot \}$



$$T_1\!: \; \top \wedge \begin{cases} x' := x \\ y' := x \end{cases}$$

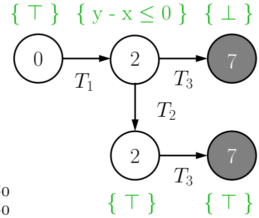
$$T_2\!: \; x \geq 1 \wedge \begin{cases} x' := x - 1 \\ y' := y - 1 \end{cases}$$

$$T_3\!: \; x \leq 0 \wedge y \geq 1 \wedge \begin{cases} x' := x \\ y' := y \end{cases}$$

Application to Program Verification

Lazy Abstraction with Interpolants

(Abstract) Program Unwinding

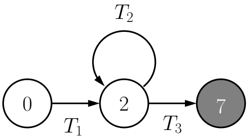


true
x$_1$ = x$_0$
y$_1$ = x$_0$

x$_1$ ≥ 1
x$_2$ = x$_1$ - 1
y$_2$ = y$_1$ - 1

x$_2$ ≤ 0
y$_2$ ≥ 1
x$_3$ = x$_2$
y$_3$ = y$_2$

Control Flow and Transitions



$$T_1\colon\ \top \wedge \begin{cases} x' := x \\ y' := x \end{cases}$$

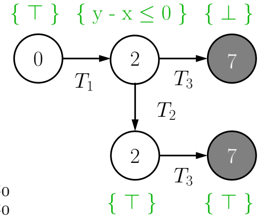
$$T_2\colon\ x \geq 1 \wedge \begin{cases} x' := x - 1 \\ y' := y - 1 \end{cases}$$

$$T_3\colon\ x \leq 0 \wedge y \geq 1 \wedge \begin{cases} x' := x \\ y' := y \end{cases}$$

Application to Program Verification

Lazy Abstraction with Interpolants

(Abstract) Program Unwinding



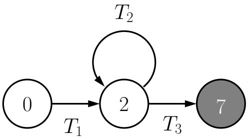
{ ⊤ }

```
true
x1 = x0
y1 = x0

x1 ≥ 1
x2 = x1 - 1
y2 = y1 - 1

x2 ≤ 0
y2 ≥ 1
x3 = x2
y3 = y2
```

Control Flow and Transitions



$$T_1\colon\ \top \wedge \begin{cases} x' := x \\ y' := x \end{cases}$$

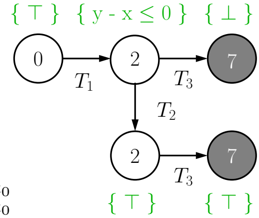
$$T_2\colon\ x \geq 1 \wedge \begin{cases} x' := x - 1 \\ y' := y - 1 \end{cases}$$

$$T_3\colon\ x \leq 0 \wedge y \geq 1 \wedge \begin{cases} x' := x \\ y' := y \end{cases}$$

Application to Program Verification

Lazy Abstraction with Interpolants

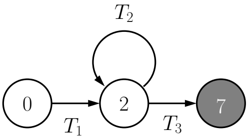
(Abstract) Program Unwinding



{ ⊤ }
true
$x_1 = x_0$
$y_1 = x_0$
{ $y_1 - x_1 \leq 0$ }
$x_1 \geq 1$
$x_2 = x_1 - 1$
$y_2 = y_1 - 1$

$x_2 \leq 0$
$y_2 \geq 1$
$x_3 = x_2$
$y_3 = y_2$

Control Flow and Transitions



$$T_1\colon\ \top \wedge \begin{cases} x' := x \\ y' := x \end{cases}$$

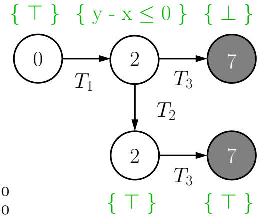
$$T_2\colon\ x \geq 1 \wedge \begin{cases} x' := x - 1 \\ y' := y - 1 \end{cases}$$

$$T_3\colon\ x \leq 0 \wedge y \geq 1 \wedge \begin{cases} x' := x \\ y' := y \end{cases}$$

Application to Program Verification

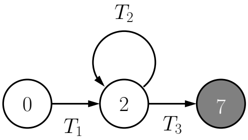
Lazy Abstraction with Interpolants

(Abstract) Program Unwinding



{ ⊤ }
  true
  $x_1 = x_0$
  $y_1 = x_0$
{ $y_1 - x_1 \leq 0$ }
  $x_1 \geq 1$
  $x_2 = x_1 - 1$
  $y_2 = y_1 - 1$
{ $y_2 - x_2 \leq 0$ }
  $x_2 \leq 0$
  $y_2 \geq 1$
  $x_3 = x_2$
  $y_3 = y_2$

Control Flow and Transitions



$$T_1\colon\ \top \wedge \begin{cases} x' := x \\ y' := x \end{cases}$$

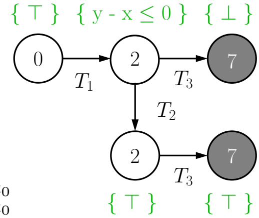
$$T_2\colon\ x \geq 1 \wedge \begin{cases} x' := x - 1 \\ y' := y - 1 \end{cases}$$

$$T_3\colon\ x \leq 0 \wedge y \geq 1 \wedge \begin{cases} x' := x \\ y' := y \end{cases}$$

Application to Program Verification

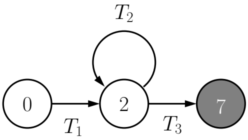
Lazy Abstraction with Interpolants

(Abstract) Program Unwinding



{ ⊤ }
  true
  $x_1 = x_0$
  $y_1 = x_0$
{ $y_1 - x_1 \leq 0$ }
  $x_1 \geq 1$
  $x_2 = x_1 - 1$
  $y_2 = y_1 - 1$
{ $y_2 - x_2 \leq 0$ }
  $x_2 \leq 0$
  $y_2 \geq 1$
  $x_3 = x_2$
  $y_3 = y_2$
{ ⊥ }

Control Flow and Transitions

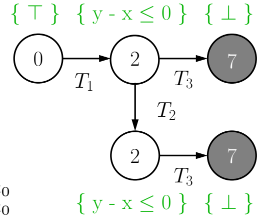


$$T_1\text{: } \top \wedge \begin{cases} x' := x \\ y' := x \end{cases}$$

$$T_2\text{: } x \geq 1 \wedge \begin{cases} x' := x - 1 \\ y' := y - 1 \end{cases}$$

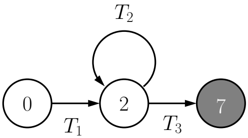
$$T_3\text{: } x \leq 0 \wedge y \geq 1 \wedge \begin{cases} x' := x \\ y' := y \end{cases}$$

(Abstract) Program Unwinding



{ $\top$ }
true
$x_1 = x_0$
$y_1 = x_0$
{ $y_1 - x_1 \leq 0$ }
$x_1 \geq 1$
$x_2 = x_1 - 1$
$y_2 = y_1 - 1$
{ $y_2 - x_2 \leq 0$ }
$x_2 \leq 0$
$y_2 \geq 1$
$x_3 = x_2$
$y_3 = y_2$
{ $\perp$ }

Control Flow and Transitions



$$T_1\colon\ \top \wedge \begin{cases} x' := x \\ y' := x \end{cases}$$

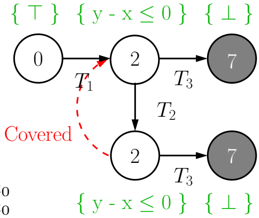
$$T_2\colon\ x \geq 1 \wedge \begin{cases} x' := x - 1 \\ y' := y - 1 \end{cases}$$

$$T_3\colon\ x \leq 0 \wedge y \geq 1 \wedge \begin{cases} x' := x \\ y' := y \end{cases}$$

Application to Program Verification

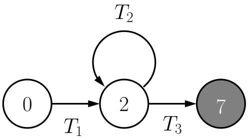
Lazy Abstraction with Interpolants

(Abstract) Program Unwinding



{ ⊤ }
  true
  $x_1 = x_0$
  $y_1 = x_0$
{ $y_1 - x_1 \leq 0$ }
  $x_1 \geq 1$
  $x_2 = x_1 - 1$
  $y_2 = y_1 - 1$
{ $y_2 - x_2 \leq 0$ }
  $x_2 \leq 0$
  $y_2 \geq 1$
  $x_3 = x_2$
  $y_3 = y_2$
{ ⊥ }

Control Flow and Transitions



$$T_1\colon\ \top \wedge \begin{cases} x' := x \\ y' := x \end{cases}$$

$$T_2\colon\ x \geq 1 \wedge \begin{cases} x' := x - 1 \\ y' := y - 1 \end{cases}$$

$$T_3\colon\ x \leq 0 \wedge y \geq 1 \wedge \begin{cases} x' := x \\ y' := y \end{cases}$$

Demo
Time

Computing Interpolants

Interpolants via Quantifier Elimination (QE)

If T admits QE, then an interpolant for $A \wedge B$ can be computed as follows:

- Take A. Let $\vec{a}$ be the symbols local to A;

Interpolants via Quantifier Elimination (QE)

If T admits QE, then an interpolant for $A \wedge B$ can be computed as follows:

- Take A. Let $\vec{a}$ be the symbols local to A;
- "Quantify-out" local symbols as $\exists \vec{a}.\ A$;

Interpolants via Quantifier Elimination (QE)

If T admits QE, then an interpolant for $A \wedge B$ can be computed as follows:

- Take A. Let $\vec{a}$ be the symbols local to A;
- "Quantify-out" local symbols as $\exists \vec{a}.\ A$;
- Compute $I = \mathrm{QE}(\exists \vec{a}.\ A)$.

Interpolants via Quantifier Elimination (QE)

If T admits QE, then an interpolant for $A \wedge B$ can be computed as follows:

- Take A. Let $\vec{a}$ be the symbols local to A;
- "Quantify-out" local symbols as $\exists \vec{a}.\ A$;
- Compute $I = \mathrm{QE}(\exists \vec{a}.\ A)$.

Interpolants via Quantifier Elimination (QE)

If T admits QE, then an interpolant for $A \wedge B$ can be computed as follows:

- Take A. Let $\vec{a}$ be the symbols local to A;
- "Quantify-out" local symbols as $\exists \vec{a}.\ A$;
- Compute $I = \mathrm{QE}(\exists \vec{a}.\ A)$.

Example (Boolean logic)

$A \equiv \{\neg a \wedge (a \vee c_1) \wedge (a \vee c_2)\}$
$B \equiv \{\neg b \wedge (b \vee \neg c_1) \wedge (b \vee \neg c_2)\}$

Interpolants via Quantifier Elimination (QE)

If T admits QE, then an interpolant for $A \wedge B$ can be computed as follows:

- Take A. Let $\vec{a}$ be the symbols local to A;
- "Quantify-out" local symbols as $\exists \vec{a}.\ A$;
- Compute $I = \text{QE}(\exists \vec{a}.\ A)$.

Example (Boolean logic)

$A \equiv \{\neg a \wedge (a \vee c_1) \wedge (a \vee c_2)\}$

$B \equiv \{\neg b \wedge (b \vee \neg c_1) \wedge (b \vee \neg c_2)\}$

$I = \text{QE}(\exists a.\ A)$

Interpolants via Quantifier Elimination (QE)

If T admits QE, then an interpolant for $A \wedge B$ can be computed as follows:

- Take A. Let $\vec{a}$ be the symbols local to A;
- "Quantify-out" local symbols as $\exists \vec{a}.\ A$;
- Compute $I = \mathrm{QE}(\exists \vec{a}.\ A)$.

Example (Boolean logic)

$A \equiv \{\neg a \wedge (a \vee c_1) \wedge (a \vee c_2)\}$
$B \equiv \{\neg b \wedge (b \vee \neg c_1) \wedge (b \vee \neg c_2)\}$
$I = \mathrm{QE}(\exists a.\ A) = A(\top / a) \vee A(\bot / a)$

Interpolants via Quantifier Elimination (QE)

If T admits QE, then an interpolant for $A \wedge B$ can be computed as follows:

- Take A. Let $\vec{a}$ be the symbols local to A;
- "Quantify-out" local symbols as $\exists \vec{a}.\ A$;
- Compute $I = \mathrm{QE}(\exists \vec{a}.\ A)$.

Example (Boolean logic)

$A \equiv \{\neg a \wedge (a \vee c_1) \wedge (a \vee c_2)\}$
$B \equiv \{\neg b \wedge (b \vee \neg c_1) \wedge (b \vee \neg c_2)\}$
$I = \mathrm{QE}(\exists a.\ A) = A(\top/a) \vee A(\bot/a) = \bot \vee (c_1 \wedge c_2)$

Interpolants via Quantifier Elimination (QE)

If T admits QE, then an interpolant for $A \wedge B$ can be computed as follows:

- Take A. Let $\vec{a}$ be the symbols local to A;
- "Quantify-out" local symbols as $\exists\vec{a}.\ A$;
- Compute $I = \mathrm{QE}(\exists\vec{a}.\ A)$.

Example (Boolean logic)

$A \equiv \{\neg a \wedge (a \vee c_1) \wedge (a \vee c_2)\}$
$B \equiv \{\neg b \wedge (b \vee \neg c_1) \wedge (b \vee \neg c_2)\}$
$I = \mathrm{QE}(\exists a.\ A) = A(\top/a) \vee A(\bot/a) = \bot \vee (c_1 \wedge c_2) = c_1 \wedge c_2$

Interpolants via Quantifier Elimination (QE)

If T admits QE, then an interpolant for $A \wedge B$ can be computed as follows:

- Take A. Let $\vec{a}$ be the symbols local to A;
- "Quantify-out" local symbols as $\exists \vec{a}.\ A$;
- Compute $I = \mathrm{QE}(\exists \vec{a}.\ A)$.

Example (Boolean logic)

$A \equiv \{\neg a \wedge (a \vee c_1) \wedge (a \vee c_2)\}$
$B \equiv \{\neg b \wedge (b \vee \neg c_1) \wedge (b \vee \neg c_2)\}$
$I = \mathrm{QE}(\exists a.\ A) = A(\top/a) \vee A(\bot/a) = \bot \vee (c_1 \wedge c_2) = c_1 \wedge c_2$

Interpolants computed this way are the "strongest" possible, as $T \vdash A \leftrightarrow I$

Interpolants via Quantifier Elimination (QE)

If T admits QE, then an interpolant for $A \wedge B$ can be computed as follows:

- Take A. Let $\vec{a}$ be the symbols local to A;
- "Quantify-out" local symbols as $\exists \vec{a}.\ A$;
- Compute $I = \mathrm{QE}(\exists \vec{a}.\ A)$.

Example (Boolean logic)

$A \equiv \{\neg a \wedge (a \vee c_1) \wedge (a \vee c_2)\}$
$B \equiv \{\neg b \wedge (b \vee \neg c_1) \wedge (b \vee \neg c_2)\}$
$I = \mathrm{QE}(\exists a.\ A) = A(\top/a) \vee A(\bot/a) = \bot \vee (c_1 \wedge c_2) = c_1 \wedge c_2$

Interpolants computed this way are the "strongest" possible, as $T \vdash A \leftrightarrow I$ (remember that by definition $T \vdash A \rightarrow I$ is enough)

Interpolants via Quantifier Elimination (QE)

If T admits QE, then an interpolant for $A \wedge B$ can be computed as follows:

- Take A. Let $\vec{a}$ be the symbols local to A;
- "Quantify-out" local symbols as $\exists \vec{a}.\ A$;
- Compute $I = \mathrm{QE}(\exists \vec{a}.\ A)$.

Example (Boolean logic)

$A \equiv \{\neg a \wedge (a \vee c_1) \wedge (a \vee c_2)\}$
$B \equiv \{\neg b \wedge (b \vee \neg c_1) \wedge (b \vee \neg c_2)\}$
$I = \mathrm{QE}(\exists a.\ A) = A(\top/a) \vee A(\bot/a) = \bot \vee (c_1 \wedge c_2) = c_1 \wedge c_2$

Interpolants computed this way are the "strongest" possible, as $T \vdash A \leftrightarrow I$ (remember that by definition $T \vdash A \rightarrow I$ is enough)

This technique is computationally **too expensive** and is to be avoided

Interpolants Computation in Practice

Several ways of describing interpolant computation:

Interpolants Computation in Practice

Several ways of describing interpolant computation:

- By extending rules of an existing calculus with a set of "interpolating instructions"

$$\dfrac{\Gamma, b_1 \vdash \Delta \qquad \Gamma, b_2 \vdash \Delta}{\Gamma, b_1 \vee b_2 \vdash \Delta}\ \vee\text{-Left}$$

Interpolants Computation in Practice

Several ways of describing interpolant computation:

- By extending rules of an existing calculus with a set of "interpolating instructions"

$$\frac{\Gamma, b_1 \vdash \Delta \parallel I_1 \qquad \Gamma, b_2 \vdash \Delta \parallel I_2}{\Gamma, b_1 \vee b_2 \vdash \Delta \parallel I_1 \wedge I_2} \vee\text{-Left}$$

Interpolants Computation in Practice

Several ways of describing interpolant computation:

- By extending rules of an existing calculus with a set of "interpolating instructions"

$$\frac{\Gamma, b_1 \vdash \Delta \parallel I_1 \qquad \Gamma, b_2 \vdash \Delta \parallel I_2}{\Gamma, b_1 \vee b_2 \vdash \Delta \parallel I_1 \wedge I_2} \vee\text{-Left}$$

  (+) Formally very clean
  (−) Non-deterministic

Interpolants Computation in Practice

Several ways of describing interpolant computation:

- By extending rules of an existing calculus with a set of "interpolating instructions"

$$\frac{\Gamma, b_1 \vdash \Delta \parallel I_1 \qquad\qquad \Gamma, b_2 \vdash \Delta \parallel I_2}{\Gamma, b_1 \vee b_2 \vdash \Delta \parallel I_1 \wedge I_2} \vee\text{-Left}$$

  (+) Formally very clean
  (−) Non-deterministic

- By extending an existing algorithm, e.g., the Simplex: output the summaries of the constraints belonging to A that are involved in the conflict

Interpolants Computation in Practice

Several ways of describing interpolant computation:

- By extending rules of an existing calculus with a set of "interpolating instructions"

$$\frac{\Gamma, b_1 \vdash \Delta \parallel I_1 \qquad \Gamma, b_2 \vdash \Delta \parallel I_2}{\Gamma, b_1 \vee b_2 \vdash \Delta \parallel I_1 \wedge I_2} \vee\text{-Left}$$

(+) Formally very clean
(−) Non-deterministic

- By extending an existing algorithm, e.g., the Simplex: output the summaries of the constraints belonging to A that are involved in the conflict

A	$x + y + z \leq 0$ $-2y + 3z \leq 0$	1 1/2
B	$-\frac{3}{5}x - \frac{3}{2}z \leq -3$	5/3

Interpolants Computation in Practice

Several ways of describing interpolant computation:

- By extending rules of an existing calculus with a set of "interpolating instructions"

$$\frac{\Gamma, b_1 \vdash \Delta \parallel I_1 \qquad \Gamma, b_2 \vdash \Delta \parallel I_2}{\Gamma, b_1 \vee b_2 \vdash \Delta \parallel I_1 \wedge I_2} \vee\text{-Left}$$

  (+) Formally very clean
  (−) Non-deterministic

- By extending an existing algorithm, e.g., the Simplex: output the summaries of the constraints belonging to A that are involved in the conflict

A	$x + y + z \leq 0$	1
	$-2y + 3z \leq 0$	1/2
I	$x + \frac{5}{2}z \leq 0$	
B	$-\frac{3}{5}x - \frac{3}{2}z \leq -3$	5/3

Interpolants Computation in Practice

Several ways of describing interpolant computation:

- By extending rules of an existing calculus with a set of "interpolating instructions"

$$\frac{\Gamma, b_1 \vdash \Delta \parallel I_1 \qquad \Gamma, b_2 \vdash \Delta \parallel I_2}{\Gamma, b_1 \vee b_2 \vdash \Delta \parallel I_1 \wedge I_2} \vee\text{-Left}$$

(+) Formally very clean
(−) Non-deterministic

- By extending an existing algorithm, e.g., the Simplex: output the summaries of the constraints belonging to A that are involved in the conflict

A	$x + y + z \leq 0$	1
	$-2y + 3z \leq 0$	1/2
I	$x + \frac{5}{2}z \leq 0$	
B	$-\frac{3}{5}x - \frac{3}{2}z \leq -3$	5/3

(+) Algorithmically precise
(−) Low flexibility

Two-Provers Paradigm

Abstract way of describing interpolation process, detached from any particular calculus or algorithm

Two-Provers Paradigm

Abstract way of describing interpolation process, detached from any particular calculus or algorithm

Two identical provers, one for A and one for B cooperate in turns to derive unsatisfiability (similarly to Nelson-Oppen framework). At any step provers either

A	B
γ_1	δ_1
γ_2	δ_2

Two-Provers Paradigm

Abstract way of describing interpolation process, detached from any particular calculus or algorithm

Two identical provers, one for A and one for B cooperate in turns to derive unsatisfiability (similarly to Nelson-Oppen framework). At any step provers either

- locally **derive** new facts

A	B
γ_1	δ_1
γ_2	δ_2

Two-Provers Paradigm

Abstract way of describing interpolation process, detached from any particular calculus or algorithm

Two identical provers, one for A and one for B cooperate in turns to derive unsatisfiability (similarly to Nelson-Oppen framework). At any step provers either

- locally **derive** new facts

A
γ_1
γ_2
γ_3

B
δ_1
δ_2

$$A \vdash \gamma_3$$

Two-Provers Paradigm

Abstract way of describing interpolation process, detached from any particular calculus or algorithm

Two identical provers, one for A and one for B cooperate in turns to derive unsatisfiability (similarly to Nelson-Oppen framework). At any step provers either

- locally **derive** new facts
- **exchange** information on the shared language with the other prover

A
γ_1
γ_2
γ_3

B
δ_1
δ_2

Two-Provers Paradigm

Abstract way of describing interpolation process, detached from any particular calculus or algorithm

Two identical provers, one for A and one for B cooperate in turns to derive unsatisfiability (similarly to Nelson-Oppen framework). At any step provers either

- locally **derive** new facts
- **exchange** information on the shared language with the other prover

A		B
γ_1		δ_1
γ_2		δ_2
γ_3	$\rightarrow$	γ_3

If γ_3 is on common language

Two-Provers Paradigm

Abstract way of describing interpolation process, detached from any particular calculus or algorithm

Two identical provers, one for A and one for B cooperate in turns to derive unsatisfiability (similarly to Nelson-Oppen framework). At any step provers either

- locally **derive** new facts
- **exchange** information on the shared language with the other prover

A
γ_1
γ_2
γ_3

B
δ_1
δ_2
γ_3

Repeat until either A or B derive $\perp$

Two-Provers Paradigm

Abstract way of describing interpolation process, detached from any particular calculus or algorithm

Two identical provers, one for A and one for B cooperate in turns to derive unsatisfiability (similarly to Nelson-Oppen framework). At any step provers either

- locally **derive** new facts
- **exchange** information on the shared language with the other prover

A
γ_1
γ_2
γ_3

B
δ_1
δ_2
γ_3

Repeat until either A or B derive $\perp$

Interpolant can be computed in backward manner

Two-Provers Paradigm

<table>
<tr><th colspan="2">Base Cases</th></tr>
<tr><td>$\begin{array}{|c|}\hline A \\ \hline \cdots \\ \perp \\ \hline \end{array} \quad \begin{array}{|c|}\hline B \\ \hline \cdots \\ \\ \hline \end{array}$</td><td></td></tr>
<tr><th colspan="2">Derivations</th></tr>
<tr><td></td><td></td></tr>
<tr><th colspan="2">Exchange</th></tr>
<tr><td></td><td></td></tr>
</table>

Two-Provers Paradigm

<table>
<tr><th colspan="2">Base Cases</th></tr>
<tr><td>$\begin{array}{|c|}\hline \color{red}{A} \\ \hline \cdots \\ \perp \\ \hline\end{array} \quad \begin{array}{|c|}\hline \color{blue}{B} \\ \hline \cdots \\ \\ \hline\end{array}$

$\color{green}{I} \equiv \perp$</td><td></td></tr>
<tr><th colspan="2">Derivations</th></tr>
<tr><td></td><td></td></tr>
<tr><th colspan="2">Exchange</th></tr>
<tr><td></td><td></td></tr>
</table>

Two-Provers Paradigm

<table>
<tr><th colspan="2">Base Cases</th></tr>
<tr><td>$\begin{array}{|c|}\hline {\color{red}A} \\ \hline \cdots \\ \perp \\ \hline\end{array} \quad \begin{array}{|c|}\hline {\color{blue}B} \\ \hline \cdots \\ \\ \hline\end{array}$

${\color{green}I} \equiv \perp$</td><td>$\begin{array}{|c|}\hline {\color{red}A} \\ \hline \cdots \\ \\ \hline\end{array} \quad \begin{array}{|c|}\hline {\color{blue}B} \\ \hline \cdots \\ \perp \\ \hline\end{array}$</td></tr>
<tr><th colspan="2">Derivations</th></tr>
<tr><td></td><td></td></tr>
<tr><th colspan="2">Exchange</th></tr>
<tr><td></td><td></td></tr>
</table>

Two-Provers Paradigm

<table>
<tr><th colspan="2">Base Cases</th></tr>
<tr><td>$\begin{array}{|c|}\hline A \\ \hline \cdots \\ \perp \\ \hline \end{array} \quad \begin{array}{|c|}\hline B \\ \hline \cdots \\ \\ \hline \end{array}$
$I \equiv \perp$</td><td>$\begin{array}{|c|}\hline A \\ \hline \cdots \\ \\ \hline \end{array} \quad \begin{array}{|c|}\hline B \\ \hline \cdots \\ \perp \\ \hline \end{array}$
$I \equiv \top$</td></tr>
<tr><th colspan="2">Derivations</th></tr>
<tr><td></td><td></td></tr>
<tr><th colspan="2">Exchange</th></tr>
<tr><td></td><td></td></tr>
</table>

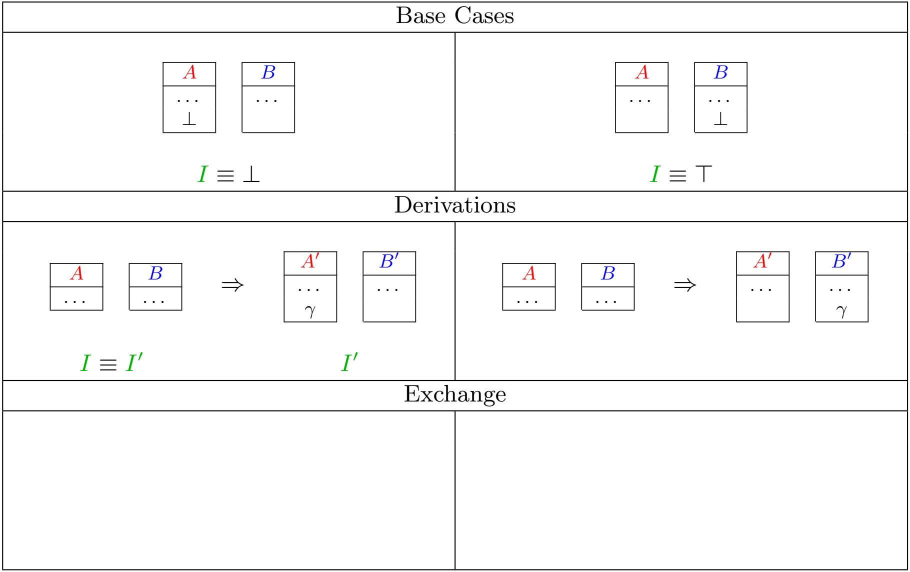
Two-Provers Paradigm

Base Cases	
$\begin{array}{|c|}\hline \color{red}{A} \\ \hline \cdots \\ \bot \\ \hline \end{array} \quad \begin{array}{|c|}\hline \color{blue}{B} \\ \hline \cdots \\ \\ \hline \end{array}$ $\color{green}{I} \equiv \bot$	$\begin{array}{|c|}\hline \color{red}{A} \\ \hline \cdots \\ \\ \hline \end{array} \quad \begin{array}{|c|}\hline \color{blue}{B} \\ \hline \cdots \\ \bot \\ \hline \end{array}$ $\color{green}{I} \equiv \top$
Derivations	
$\begin{array}{|c|}\hline \color{red}{A} \\ \hline \cdots \\ \hline \end{array} \quad \begin{array}{|c|}\hline \color{blue}{B} \\ \hline \cdots \\ \hline \end{array} \quad \Rightarrow \quad \begin{array}{|c|}\hline \color{red}{A'} \\ \hline \cdots \\ \gamma \\ \hline \end{array} \quad \begin{array}{|c|}\hline \color{blue}{B'} \\ \hline \cdots \\ \\ \hline \end{array}$	
Exchange	

Two-Provers Paradigm

Base Cases	
$\begin{array}{\|c\|}\hline {\color{red}A} \\ \hline \cdots \\ \bot \\ \hline\end{array} \quad \begin{array}{\|c\|}\hline {\color{blue}B} \\ \hline \cdots \\ \\ \hline\end{array}$ ${\color{green}I} \equiv \bot$	$\begin{array}{\|c\|}\hline {\color{red}A} \\ \hline \cdots \\ \\ \hline\end{array} \quad \begin{array}{\|c\|}\hline {\color{blue}B} \\ \hline \cdots \\ \bot \\ \hline\end{array}$ ${\color{green}I} \equiv \top$
Derivations	
$\begin{array}{\|c\|}\hline {\color{red}A} \\ \hline \cdots \\ \hline\end{array} \quad \begin{array}{\|c\|}\hline {\color{blue}B} \\ \hline \cdots \\ \hline\end{array} \quad \Rightarrow \quad \begin{array}{\|c\|}\hline {\color{red}A'} \\ \hline \cdots \\ \gamma \\ \hline\end{array} \quad \begin{array}{\|c\|}\hline {\color{blue}B'} \\ \hline \cdots \\ \\ \hline\end{array}$ ${\color{green}I} \equiv {\color{green}I'}$ $\qquad$ ${\color{green}I'}$	
Exchange	

Two-Provers Paradigm



Two-Provers Paradigm

Base Cases	
$\begin{array}{\|c\|}\hline A \\ \hline \cdots \\ \bot \\ \hline\end{array} \quad \begin{array}{\|c\|}\hline B \\ \hline \cdots \\ \\ \hline\end{array}$ $I \equiv \bot$	$\begin{array}{\|c\|}\hline A \\ \hline \cdots \\ \\ \hline\end{array} \quad \begin{array}{\|c\|}\hline B \\ \hline \cdots \\ \bot \\ \hline\end{array}$ $I \equiv \top$
Derivations	
$\begin{array}{\|c\|}\hline A \\ \hline \cdots \\ \hline\end{array} \quad \begin{array}{\|c\|}\hline B \\ \hline \cdots \\ \hline\end{array} \quad \Rightarrow \quad \begin{array}{\|c\|}\hline A' \\ \hline \cdots \\ \gamma \\ \hline\end{array} \quad \begin{array}{\|c\|}\hline B' \\ \hline \cdots \\ \\ \hline\end{array}$ $I \equiv I'$ $\qquad$ I'	$\begin{array}{\|c\|}\hline A \\ \hline \cdots \\ \hline\end{array} \quad \begin{array}{\|c\|}\hline B \\ \hline \cdots \\ \hline\end{array} \quad \Rightarrow \quad \begin{array}{\|c\|}\hline A' \\ \hline \cdots \\ \\ \hline\end{array} \quad \begin{array}{\|c\|}\hline B' \\ \hline \cdots \\ \gamma \\ \hline\end{array}$ $I \equiv I'$ $\qquad$ I'
Exchange	

Two-Provers Paradigm

Base Cases	
$\begin{array}{\|c\|}\hline \color{red}{A} \\ \hline \cdots \\ \bot \\ \hline\end{array} \quad \begin{array}{\|c\|}\hline \color{blue}{B} \\ \hline \cdots \\ \\ \hline\end{array}$ $\color{green}{I} \equiv \bot$	$\begin{array}{\|c\|}\hline \color{red}{A} \\ \hline \cdots \\ \\ \hline\end{array} \quad \begin{array}{\|c\|}\hline \color{blue}{B} \\ \hline \cdots \\ \bot \\ \hline\end{array}$ $\color{green}{I} \equiv \top$
Derivations	
$\begin{array}{\|c\|}\hline \color{red}{A} \\ \hline \cdots \\ \hline\end{array} \quad \begin{array}{\|c\|}\hline \color{blue}{B} \\ \hline \cdots \\ \hline\end{array} \quad \Rightarrow \quad \begin{array}{\|c\|}\hline \color{red}{A'} \\ \hline \cdots \\ \gamma \\ \hline\end{array} \quad \begin{array}{\|c\|}\hline \color{blue}{B'} \\ \hline \cdots \\ \\ \hline\end{array}$ $\color{green}{I} \equiv \color{green}{I'} \qquad \color{green}{I'}$	$\begin{array}{\|c\|}\hline \color{red}{A} \\ \hline \cdots \\ \hline\end{array} \quad \begin{array}{\|c\|}\hline \color{blue}{B} \\ \hline \cdots \\ \hline\end{array} \quad \Rightarrow \quad \begin{array}{\|c\|}\hline \color{red}{A'} \\ \hline \cdots \\ \\ \hline\end{array} \quad \begin{array}{\|c\|}\hline \color{blue}{B'} \\ \hline \cdots \\ \gamma \\ \hline\end{array}$ $\color{green}{I} \equiv \color{green}{I'} \qquad \color{green}{I'}$
Exchange	
$\begin{array}{\|c\|}\hline \color{red}{A} \\ \hline \cdots \\ \gamma \\ \hline\end{array} \quad \begin{array}{\|c\|}\hline \color{blue}{B} \\ \hline \cdots \\ \\ \hline\end{array} \quad \Rightarrow \quad \begin{array}{\|c\|}\hline \color{red}{A'} \\ \hline \cdots \\ \gamma \\ \hline\end{array} \quad \begin{array}{\|c\|}\hline \color{blue}{B'} \\ \hline \cdots \\ \gamma \\ \hline\end{array}$	

Two-Provers Paradigm

<table>
<tr><th colspan="2">Base Cases</th></tr>
<tr>
<td>$\begin{array}{|c|}\hline A \\ \hline \cdots \\ \bot \\ \hline\end{array} \quad \begin{array}{|c|}\hline B \\ \hline \cdots \\ \\ \hline\end{array}$
$I \equiv \bot$</td>
<td>$\begin{array}{|c|}\hline A \\ \hline \cdots \\ \\ \hline\end{array} \quad \begin{array}{|c|}\hline B \\ \hline \cdots \\ \bot \\ \hline\end{array}$
$I \equiv \top$</td>
</tr>
<tr><th colspan="2">Derivations</th></tr>
<tr>
<td>$\begin{array}{|c|}\hline A \\ \hline \cdots \\ \hline\end{array} \quad \begin{array}{|c|}\hline B \\ \hline \cdots \\ \hline\end{array} \quad \Rightarrow \quad \begin{array}{|c|}\hline A' \\ \hline \cdots \\ \gamma \\ \hline\end{array} \quad \begin{array}{|c|}\hline B' \\ \hline \cdots \\ \\ \hline\end{array}$
$I \equiv I' \qquad I'$</td>
<td>$\begin{array}{|c|}\hline A \\ \hline \cdots \\ \hline\end{array} \quad \begin{array}{|c|}\hline B \\ \hline \cdots \\ \hline\end{array} \quad \Rightarrow \quad \begin{array}{|c|}\hline A' \\ \hline \cdots \\ \\ \hline\end{array} \quad \begin{array}{|c|}\hline B' \\ \hline \cdots \\ \gamma \\ \hline\end{array}$
$I \equiv I' \qquad I'$</td>
</tr>
<tr><th colspan="2">Exchange</th></tr>
<tr>
<td>$\begin{array}{|c|}\hline A \\ \hline \cdots \\ \gamma \\ \hline\end{array} \quad \begin{array}{|c|}\hline B \\ \hline \cdots \\ \\ \hline\end{array} \quad \Rightarrow \quad \begin{array}{|c|}\hline A' \\ \hline \cdots \\ \gamma \\ \hline\end{array} \quad \begin{array}{|c|}\hline B' \\ \hline \cdots \\ \gamma \\ \hline\end{array}$
$I \equiv \gamma \wedge I' \qquad I'$</td>
<td></td>
</tr>
</table>

Two-Provers Paradigm

Base Cases	
$\begin{array}{\|c\|}\hline A \\ \hline \cdots \\ \bot \\ \hline\end{array} \quad \begin{array}{\|c\|}\hline B \\ \hline \cdots \\ \\ \hline\end{array}$ $I \equiv \bot$	$\begin{array}{\|c\|}\hline A \\ \hline \cdots \\ \\ \hline\end{array} \quad \begin{array}{\|c\|}\hline B \\ \hline \cdots \\ \bot \\ \hline\end{array}$ $I \equiv \top$
Derivations	
$\begin{array}{\|c\|}\hline A \\ \hline \cdots \\ \hline\end{array} \quad \begin{array}{\|c\|}\hline B \\ \hline \cdots \\ \hline\end{array} \quad \Rightarrow \quad \begin{array}{\|c\|}\hline A' \\ \hline \cdots \\ \gamma \\ \hline\end{array} \quad \begin{array}{\|c\|}\hline B' \\ \hline \cdots \\ \\ \hline\end{array}$ $I \equiv I' \qquad\qquad I'$	$\begin{array}{\|c\|}\hline A \\ \hline \cdots \\ \hline\end{array} \quad \begin{array}{\|c\|}\hline B \\ \hline \cdots \\ \hline\end{array} \quad \Rightarrow \quad \begin{array}{\|c\|}\hline A' \\ \hline \cdots \\ \\ \hline\end{array} \quad \begin{array}{\|c\|}\hline B' \\ \hline \cdots \\ \gamma \\ \hline\end{array}$ $I \equiv I' \qquad\qquad I'$
Exchange	
$\begin{array}{\|c\|}\hline A \\ \hline \cdots \\ \gamma \\ \hline\end{array} \quad \begin{array}{\|c\|}\hline B \\ \hline \cdots \\ \\ \hline\end{array} \quad \Rightarrow \quad \begin{array}{\|c\|}\hline A' \\ \hline \cdots \\ \gamma \\ \hline\end{array} \quad \begin{array}{\|c\|}\hline B' \\ \hline \cdots \\ \gamma \\ \hline\end{array}$ $I \equiv \gamma \wedge I' \qquad\qquad I'$	$\begin{array}{\|c\|}\hline A \\ \hline \cdots \\ \\ \hline\end{array} \quad \begin{array}{\|c\|}\hline B \\ \hline \cdots \\ \gamma \\ \hline\end{array} \quad \Rightarrow \quad \begin{array}{\|c\|}\hline A' \\ \hline \cdots \\ \gamma \\ \hline\end{array} \quad \begin{array}{\|c\|}\hline B' \\ \hline \cdots \\ \gamma \\ \hline\end{array}$

Two-Provers Paradigm

<table>
<tr><th colspan="2">Base Cases</th></tr>
<tr>
<td>$\begin{array}{|c|}\hline A \\ \hline \dots \\ \bot \\ \hline\end{array} \quad \begin{array}{|c|}\hline B \\ \hline \dots \\ \\ \hline\end{array}$
$I \equiv \bot$</td>
<td>$\begin{array}{|c|}\hline A \\ \hline \dots \\ \\ \hline\end{array} \quad \begin{array}{|c|}\hline B \\ \hline \dots \\ \bot \\ \hline\end{array}$
$I \equiv \top$</td>
</tr>
<tr><th colspan="2">Derivations</th></tr>
<tr>
<td>$\begin{array}{|c|}\hline A \\ \hline \dots \\ \hline\end{array} \quad \begin{array}{|c|}\hline B \\ \hline \dots \\ \hline\end{array} \quad \Rightarrow \quad \begin{array}{|c|}\hline A' \\ \hline \dots \\ \gamma \\ \hline\end{array} \quad \begin{array}{|c|}\hline B' \\ \hline \dots \\ \\ \hline\end{array}$
$I \equiv I'$ I'</td>
<td>$\begin{array}{|c|}\hline A \\ \hline \dots \\ \hline\end{array} \quad \begin{array}{|c|}\hline B \\ \hline \dots \\ \hline\end{array} \quad \Rightarrow \quad \begin{array}{|c|}\hline A' \\ \hline \dots \\ \\ \hline\end{array} \quad \begin{array}{|c|}\hline B' \\ \hline \dots \\ \gamma \\ \hline\end{array}$
$I \equiv I'$ I'</td>
</tr>
<tr><th colspan="2">Exchange</th></tr>
<tr>
<td>$\begin{array}{|c|}\hline A \\ \hline \dots \\ \gamma \\ \hline\end{array} \quad \begin{array}{|c|}\hline B \\ \hline \dots \\ \\ \hline\end{array} \quad \Rightarrow \quad \begin{array}{|c|}\hline A' \\ \hline \dots \\ \gamma \\ \hline\end{array} \quad \begin{array}{|c|}\hline B' \\ \hline \dots \\ \gamma \\ \hline\end{array}$
$I \equiv \gamma \wedge I'$ I'</td>
<td>$\begin{array}{|c|}\hline A \\ \hline \dots \\ \\ \hline\end{array} \quad \begin{array}{|c|}\hline B \\ \hline \dots \\ \gamma \\ \hline\end{array} \quad \Rightarrow \quad \begin{array}{|c|}\hline A' \\ \hline \dots \\ \gamma \\ \hline\end{array} \quad \begin{array}{|c|}\hline B' \\ \hline \dots \\ \gamma \\ \hline\end{array}$
$I \equiv \gamma \rightarrow I'$ I'</td>
</tr>
</table>

Example - Strategy 1

A
$a \vee c_1$
$a \vee c_2$
$\neg a$

B
$\neg c_1 \vee b$
$\neg c_2 \vee b$
$\neg b$

Example - Strategy 1

A
$a \vee c_1$
$a \vee c_2$
$\neg a$

B
$\neg c_1 \vee b$
$\neg c_2 \vee b$
$\neg b$

$$\xRightarrow{\text{Der.}}$$

A'
$a \vee c_1$
$a \vee c_2$
$\neg a$
c_1

B'
$\neg c_1 \vee b$
$\neg c_2 \vee b$
$\neg b$

Example - Strategy 1

A	B
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$

$\xrightarrow{\text{Der.}}$

A'	B'
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$
c_1	

$\xrightarrow{\text{Ex.}}$

A''	B''
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$
c_1	c_1

Example - Strategy 1

A	B
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$

$\xRightarrow{\text{Der.}}$

A'	B'
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$
c_1	

$\xRightarrow{\text{Ex.}}$

A''	B''
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$
c_1	c_1

$\xRightarrow{\text{Der.}^*}$

A'''	B'''
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$
c_1	c_1
	$\perp$

Example - Strategy 1

$$\begin{array}{|c|}\hline A \\ \hline a \lor c_1 \\ a \lor c_2 \\ \neg a \\ \hline\end{array} \quad \begin{array}{|c|}\hline B \\ \hline \neg c_1 \lor b \\ \neg c_2 \lor b \\ \neg b \\ \hline\end{array} \quad \overset{\text{Der.}}{\Longrightarrow} \quad \begin{array}{|c|}\hline A' \\ \hline a \lor c_1 \\ a \lor c_2 \\ \neg a \\ c_1 \\ \hline\end{array} \quad \begin{array}{|c|}\hline B' \\ \hline \neg c_1 \lor b \\ \neg c_2 \lor b \\ \neg b \\ \\ \hline\end{array} \quad \overset{\text{Ex.}}{\Longrightarrow} \quad \begin{array}{|c|}\hline A'' \\ \hline a \lor c_1 \\ a \lor c_2 \\ \neg a \\ c_1 \\ \hline\end{array} \quad \begin{array}{|c|}\hline B'' \\ \hline \neg c_1 \lor b \\ \neg c_2 \lor b \\ \neg b \\ c_1 \\ \hline\end{array}$$

$$\overset{\text{Der.}^*}{\Longrightarrow} \quad \begin{array}{|c|}\hline A''' \\ \hline a \lor c_1 \\ a \lor c_2 \\ \neg a \\ c_1 \\ \\ \hline\end{array} \quad \begin{array}{|c|}\hline B''' \\ \hline \neg c_1 \lor b \\ \neg c_2 \lor b \\ \neg b \\ c_1 \\ \bot \\ \hline\end{array}$$

$$I''' \equiv \top$$

Example - Strategy 1

A	B
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$

$\xRightarrow{\text{Der.}}$

A'	B'
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$
c_1	

$\xRightarrow{\text{Ex.}}$

A''	B''
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$
c_1	c_1

$$I'' \equiv \top$$

$\xRightarrow{\text{Der.}^*}$

A'''	B'''
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$
c_1	c_1
	$\bot$

$$I''' \equiv \top$$

Example - Strategy 1

A	B
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$

$\xRightarrow{\text{Der.}}$

A'	B'
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$
c_1	

$$I' \equiv c_1 \wedge \top$$

$\xRightarrow{\text{Ex.}}$

A''	B''
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$
c_1	c_1

$$I'' \equiv \top$$

$\xRightarrow{\text{Der.}^*}$

A'''	B'''
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$
c_1	c_1
	$\perp$

$$I''' \equiv \top$$

Example - Strategy 1

A	B
$a \lor c_1$	$\neg c_1 \lor b$
$a \lor c_2$	$\neg c_2 \lor b$
$\neg a$	$\neg b$

$I \equiv c_1 \land \top$

$\overset{\text{Der.}}{\Longrightarrow}$

A'	B'
$a \lor c_1$	$\neg c_1 \lor b$
$a \lor c_2$	$\neg c_2 \lor b$
$\neg a$	$\neg b$
c_1	

$I' \equiv c_1 \land \top$

$\overset{\text{Ex.}}{\Longrightarrow}$

A''	B''
$a \lor c_1$	$\neg c_1 \lor b$
$a \lor c_2$	$\neg c_2 \lor b$
$\neg a$	$\neg b$
c_1	c_1

$I'' \equiv \top$

$\overset{\text{Der.}^*}{\Longrightarrow}$

A'''	B'''
$a \lor c_1$	$\neg c_1 \lor b$
$a \lor c_2$	$\neg c_2 \lor b$
$\neg a$	$\neg b$
c_1	c_1
	$\bot$

$I''' \equiv \top$

Example - Strategy 1

A
$a \vee c_1$
$a \vee c_2$
$\neg a$

B
$\neg c_1 \vee b$
$\neg c_2 \vee b$
$\neg b$

$$I \equiv c_1 \wedge \top \equiv c_1$$

$$\xRightarrow{\text{Der.}}$$

A'
$a \vee c_1$
$a \vee c_2$
$\neg a$
c_1

B'
$\neg c_1 \vee b$
$\neg c_2 \vee b$
$\neg b$

$$I' \equiv c_1 \wedge \top$$

$$\xRightarrow{\text{Ex.}}$$

A''
$a \vee c_1$
$a \vee c_2$
$\neg a$
c_1

B''
$\neg c_1 \vee b$
$\neg c_2 \vee b$
$\neg b$
c_1

$$I'' \equiv \top$$

$$\xRightarrow{\text{Der.}^*}$$

A'''
$a \vee c_1$
$a \vee c_2$
$\neg a$
c_1

B'''
$\neg c_1 \vee b$
$\neg c_2 \vee b$
$\neg b$
c_1
$\bot$

$$I''' \equiv \top$$

Example - Strategy 2 (Strong interpolant)

A
$a \vee c_1$
$a \vee c_2$
$\neg a$

B
$\neg c_1 \vee b$
$\neg c_2 \vee b$
$\neg b$

Example - Strategy 2 (Strong interpolant)

A
$a \vee c_1$
$a \vee c_2$
$\neg a$

B
$\neg c_1 \vee b$
$\neg c_2 \vee b$
$\neg b$

$$\overset{\text{Der.}^*}{\Longrightarrow}$$

A'
$a \vee c_1$
$a \vee c_2$
$\neg a$
c_1
c_2

B'
$\neg c_1 \vee b$
$\neg c_2 \vee b$
$\neg b$

Example - Strategy 2 (Strong interpolant)

A
$a \vee c_1$
$a \vee c_2$
$\neg a$

B
$\neg c_1 \vee b$
$\neg c_2 \vee b$
$\neg b$

$\xRightarrow{\text{Der.}^*}$

A'
$a \vee c_1$
$a \vee c_2$
$\neg a$
c_1
c_2

B'
$\neg c_1 \vee b$
$\neg c_2 \vee b$
$\neg b$

$\xRightarrow{\text{Ex.}}$

A''
$a \vee c_1$
$a \vee c_2$
$\neg a$
c_1
c_2

B''
$\neg c_1 \vee b$
$\neg c_2 \vee b$
$\neg b$
c_1
c_2

Example - Strategy 2 (Strong interpolant)

A
$a \vee c_1$
$a \vee c_2$
$\neg a$

B
$\neg c_1 \vee b$
$\neg c_2 \vee b$
$\neg b$

$\xRightarrow{\text{Der.}^*}$

A'
$a \vee c_1$
$a \vee c_2$
$\neg a$
c_1
c_2

B'
$\neg c_1 \vee b$
$\neg c_2 \vee b$
$\neg b$

$\xRightarrow{\text{Ex.}}$

A''
$a \vee c_1$
$a \vee c_2$
$\neg a$
c_1
c_2

B''
$\neg c_1 \vee b$
$\neg c_2 \vee b$
$\neg b$
c_1
c_2

$\xRightarrow{\text{Der.}^*}$

A'''
$a \vee c_1$
$a \vee c_2$
$\neg a$
c_1
c_2

B'''
$\neg c_1 \vee b$
$\neg c_2 \vee b$
$\neg b$
c_1
c_2
$\bot$

Example - Strategy 2 (Strong interpolant)

A	B
$a \lor c_1$	$\neg c_1 \lor b$
$a \lor c_2$	$\neg c_2 \lor b$
$\neg a$	$\neg b$

$\overset{\text{Der.}^*}{\Longrightarrow}$

A'	B'
$a \lor c_1$	$\neg c_1 \lor b$
$a \lor c_2$	$\neg c_2 \lor b$
$\neg a$	$\neg b$
c_1	
c_2	

$\overset{\text{Ex.}}{\Longrightarrow}$

A''	B''
$a \lor c_1$	$\neg c_1 \lor b$
$a \lor c_2$	$\neg c_2 \lor b$
$\neg a$	$\neg b$
c_1	c_1
c_2	c_2

$\overset{\text{Der.}^*}{\Longrightarrow}$

A'''	B'''
$a \lor c_1$	$\neg c_1 \lor b$
$a \lor c_2$	$\neg c_2 \lor b$
$\neg a$	$\neg b$
c_1	c_1
c_2	c_2
	$\bot$

$I''' \equiv \top$

Example - Strategy 2 (Strong interpolant)

A
$a \vee c_1$
$a \vee c_2$
$\neg a$

B
$\neg c_1 \vee b$
$\neg c_2 \vee b$
$\neg b$

$\overset{\text{Der.}^*}{\Longrightarrow}$

A'
$a \vee c_1$
$a \vee c_2$
$\neg a$
c_1
c_2

B'
$\neg c_1 \vee b$
$\neg c_2 \vee b$
$\neg b$

$\overset{\text{Ex.}}{\Longrightarrow}$

A''
$a \vee c_1$
$a \vee c_2$
$\neg a$
c_1
c_2

B''
$\neg c_1 \vee b$
$\neg c_2 \vee b$
$\neg b$
c_1
c_2

$I'' \equiv \top$

$\overset{\text{Der.}^*}{\Longrightarrow}$

A'''
$a \vee c_1$
$a \vee c_2$
$\neg a$
c_1
c_2

B'''
$\neg c_1 \vee b$
$\neg c_2 \vee b$
$\neg b$
c_1
c_2
$\bot$

$I''' \equiv \top$

Example - Strategy 2 (Strong interpolant)

A	B
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$

$\xRightarrow{\text{Der.}^*}$

A'	B'
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$
c_1	
c_2	

$$I' \equiv c_1 \wedge c_2 \wedge \top$$

$\xRightarrow{\text{Ex.}}$

A''	B''
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$
c_1	c_1
c_2	c_2

$$I'' \equiv \top$$

$\xRightarrow{\text{Der.}^*}$

A'''	B'''
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$
c_1	c_1
c_2	c_2
	$\perp$

$$I''' \equiv \top$$

Example - Strategy 2 (Strong interpolant)

A	B
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$

$\xRightarrow{\text{Der.}^*}$

A'	B'
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$
c_1	
c_2	

$I \equiv c_1 \wedge c_2 \wedge \top$

$I' \equiv c_1 \wedge c_2 \wedge \top$

$\xRightarrow{\text{Ex.}}$

A''	B''
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$
c_1	c_1
c_2	c_2

$\xRightarrow{\text{Der.}^*}$

A'''	B'''
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$
c_1	c_1
c_2	c_2
	$\perp$

$I'' \equiv \top$

$I''' \equiv \top$

Example - Strategy 2 (Strong interpolant)

A	B
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$

$\xRightarrow{\text{Der.}^*}$

A'	B'
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$
c_1	
c_2	

$$I \equiv c_1 \wedge c_2 \wedge \top \equiv c_1 \wedge c_2$$

$$I' \equiv c_1 \wedge c_2 \wedge \top$$

$\xRightarrow{\text{Ex.}}$

A''	B''
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$
c_1	c_1
c_2	c_2

$\xRightarrow{\text{Der.}^*}$

A'''	B'''
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$
c_1	c_1
c_2	c_2
	$\perp$

$$I'' \equiv \top$$

$$I''' \equiv \top$$

Example - Strategy 3 (weak interpolant)

A
$a \vee c_1$
$a \vee c_2$
$\neg a$

B
$\neg c_1 \vee b$
$\neg c_2 \vee b$
$\neg b$

Example - Strategy 3 (weak interpolant)

A
$a \vee c_1$
$a \vee c_2$
$\neg a$

B
$\neg c_1 \vee b$
$\neg c_2 \vee b$
$\neg b$

$$\xRightarrow{\text{Der.}^*}$$

A'
$a \vee c_1$
$a \vee c_2$
$\neg a$

B'
$\neg c_1 \vee b$
$\neg c_2 \vee b$
$\neg b$
$\neg c_1$
$\neg c_2$

Example - Strategy 3 (weak interpolant)

A	B
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$

$\xRightarrow{\text{Der.}^*}$

A'	B'
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$
	$\neg c_1$
	$\neg c_2$

$\xRightarrow{\text{Ex.}}$

A''	B''
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$
$\neg c_1$	$\neg c_1$
$\neg c_2$	$\neg c_2$

Example - Strategy 3 (weak interpolant)

A
$a \vee c_1$
$a \vee c_2$
$\neg a$

B
$\neg c_1 \vee b$
$\neg c_2 \vee b$
$\neg b$

$\xRightarrow{\text{Der.}^*}$

A'
$a \vee c_1$
$a \vee c_2$
$\neg a$

B'
$\neg c_1 \vee b$
$\neg c_2 \vee b$
$\neg b$
$\neg c_1$
$\neg c_2$

$\xRightarrow{\text{Ex.}}$

A''
$a \vee c_1$
$a \vee c_2$
$\neg a$
$\neg c_1$
$\neg c_2$

B''
$\neg c_1 \vee b$
$\neg c_2 \vee b$
$\neg b$
$\neg c_1$
$\neg c_2$

$\xRightarrow{\text{Der.}^*}$

A'''
$a \vee c_1$
$a \vee c_2$
$\neg a$
$\neg c_1$
$\neg c_2$
$\perp$

B'''
$\neg c_1 \vee b$
$\neg c_2 \vee b$
$\neg b$
$\neg c_1$
$\neg c_2$

Example - Strategy 3 (weak interpolant)

A	B
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$

$\overset{\text{Der.}^*}{\Longrightarrow}$

A'	B'
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$
	$\neg c_1$
	$\neg c_2$

$\overset{\text{Ex.}}{\Longrightarrow}$

A''	B''
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$
$\neg c_1$	$\neg c_1$
$\neg c_2$	$\neg c_2$

$\overset{\text{Der.}^*}{\Longrightarrow}$

A'''	B'''
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$
$\neg c_1$	$\neg c_1$
$\neg c_2$	$\neg c_2$
$\bot$	

$$I''' \equiv \bot$$

Example - Strategy 3 (weak interpolant)

A	B
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$

$\xRightarrow{\text{Der.}^*}$

A'	B'
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$
	$\neg c_1$
	$\neg c_2$

$\xRightarrow{\text{Ex.}}$

A''	B''
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$
$\neg c_1$	$\neg c_1$
$\neg c_2$	$\neg c_2$

$I'' \equiv \perp$

$\xRightarrow{\text{Der.}^*}$

A'''	B'''
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$
$\neg c_1$	$\neg c_1$
$\neg c_2$	$\neg c_2$
$\perp$	

$I''' \equiv \perp$

Example - Strategy 3 (weak interpolant)

A	B
$a \lor c_1$	$\neg c_1 \lor b$
$a \lor c_2$	$\neg c_2 \lor b$
$\neg a$	$\neg b$

$\overset{\text{Der.}^*}{\Longrightarrow}$

A'	B'
$a \lor c_1$	$\neg c_1 \lor b$
$a \lor c_2$	$\neg c_2 \lor b$
$\neg a$	$\neg b$
	$\neg c_1$
	$\neg c_2$

$$I' \equiv (\neg c_1 \land \neg c_2) \to \bot$$

$\overset{\text{Ex.}}{\Longrightarrow}$

A''	B''
$a \lor c_1$	$\neg c_1 \lor b$
$a \lor c_2$	$\neg c_2 \lor b$
$\neg a$	$\neg b$
$\neg c_1$	$\neg c_1$
$\neg c_2$	$\neg c_2$

$$I'' \equiv \bot$$

$\overset{\text{Der.}^*}{\Longrightarrow}$

A'''	B'''
$a \lor c_1$	$\neg c_1 \lor b$
$a \lor c_2$	$\neg c_2 \lor b$
$\neg a$	$\neg b$
$\neg c_1$	$\neg c_1$
$\neg c_2$	$\neg c_2$
$\bot$	

$$I''' \equiv \bot$$

Example - Strategy 3 (weak interpolant)

A	B
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$

$$I \equiv (\neg c_1 \wedge \neg c_2) \rightarrow \bot$$

$\xRightarrow{\text{Der.}^*}$

A'	B'
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$
	$\neg c_1$
	$\neg c_2$

$$I' \equiv (\neg c_1 \wedge \neg c_2) \rightarrow \bot$$

$\xRightarrow{\text{Ex.}}$

A''	B''
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$
$\neg c_1$	$\neg c_1$
$\neg c_2$	$\neg c_2$

$$I'' \equiv \bot$$

$\xRightarrow{\text{Der.}^*}$

A'''	B'''
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$
$\neg c_1$	$\neg c_1$
$\neg c_2$	$\neg c_2$
$\bot$	

$$I''' \equiv \bot$$

Example - Strategy 3 (weak interpolant)

A	B
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$

$$\overset{\text{Der.}^*}{\Longrightarrow}$$

A'	B'
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$
	$\neg c_1$
	$\neg c_2$

$$I \equiv (\neg c_1 \wedge \neg c_2) \rightarrow \bot \equiv c_1 \vee c_2$$

$$I' \equiv (\neg c_1 \wedge \neg c_2) \rightarrow \bot$$

$$\overset{\text{Ex.}}{\Longrightarrow}$$

A''	B''
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$
$\neg c_1$	$\neg c_1$
$\neg c_2$	$\neg c_2$

$$\overset{\text{Der.}^*}{\Longrightarrow}$$

A'''	B'''
$a \vee c_1$	$\neg c_1 \vee b$
$a \vee c_2$	$\neg c_2 \vee b$
$\neg a$	$\neg b$
$\neg c_1$	$\neg c_1$
$\neg c_2$	$\neg c_2$
$\bot$	

$$I'' \equiv \bot$$

$$I''' \equiv \bot$$

Proof Transformation
(for interpolation and reduction)

- Resolution proofs find application in several ambits

- Resolution proofs find application in several ambits
  - Interpolation-based model checking
  - Abstraction techniques
  - Unsatisfiable core extraction in SAT/SMT
  - Automatic theorem proving

- Resolution proofs find application in several ambits
  - Interpolation-based model checking
  - Abstraction techniques
  - Unsatisfiable core extraction in SAT/SMT
  - Automatic theorem proving
- Problems

Proof Transformation and Reduction

Motivation

- Resolution proofs find application in several ambits
  - Interpolation-based model checking
  - Abstraction techniques
  - Unsatisfiable core extraction in SAT/SMT
  - Automatic theorem proving
- Problems
  - Clean structure of proofs is required for interpolation generation

Proof Transformation and Reduction

Motivation

- Resolution proofs find application in several ambits
  - Interpolation-based model checking
  - Abstraction techniques
  - Unsatisfiable core extraction in SAT/SMT
  - Automatic theorem proving
- Problems
  - Clean structure of proofs is required for interpolation generation
  - Size affects efficiency
  - Size can be exponential w.r.t. input formula

- Interpolant I for unsatisfiable conjunction of formulae $A \wedge B$

- Interpolant I for unsatisfiable conjunction of formulae $A \wedge B$
- State-of-the-art approach [Pudlák97, McMillan04]

- Interpolant I for unsatisfiable conjunction of formulae $A \wedge B$
- State-of-the-art approach [Pudlák97, McMillan04]
    - Derivation of unsatisfiability resolution proof of $A \wedge B$

- Interpolant I for unsatisfiable conjunction of formulae $A \wedge B$
- State-of-the-art approach [Pudlák97, McMillan04]
  - Derivation of unsatisfiability resolution proof of $A \wedge B$
  - Computation of I from proof structure in linear time

Resolution System

Background

- Literal $\quad p \quad \overline{p}$

Resolution System
Background

- Literal $\quad p \quad \overline{p}$
- Clause $\quad p \vee \overline{q} \vee r \vee \ldots \to p\overline{q}r\ldots \qquad$ Empty clause $\quad \perp$

Resolution System

Background

- Literal $p \quad \overline{p}$
- Clause $p \vee \overline{q} \vee r \vee \ldots \rightarrow p\overline{q}r \ldots$ Empty clause $\perp$
- Input formula $(p \vee q) \wedge (r \vee \overline{p}) \ldots \rightarrow \{pq, r\overline{p}\}$

Resolution System

Background

- Literal $p \quad \overline{p}$

- Clause $p \vee \overline{q} \vee r \vee \ldots \rightarrow p\overline{q}r\ldots$ Empty clause $\perp$

- Input formula $(p \vee q) \wedge (r \vee \overline{p}) \ldots \rightarrow \{pq, r\overline{p}\}$

- Resolution rule $$\frac{pC \qquad \overline{p}D}{CD}\, p$$

  Antecedents: $pC\ \overline{p}D$ Resolvent: CD Pivot: p

Resolution System
Background

- Literal $p \quad \overline{p}$
- Clause $p \vee \overline{q} \vee r \vee \ldots \;\rightarrow\; p\overline{q}r \ldots$ Empty clause $\perp$
- Input formula $(p \vee q) \wedge (r \vee \overline{p}) \ldots \;\rightarrow\; \{pq, r\overline{p}\}$
- Resolution rule $$\frac{pC \qquad\qquad \overline{p}D}{CD}\, p$$

  Antecedents: $pC\ \overline{p}D$ Resolvent: CD Pivot: p
- Resolution proof of unsatisfiability of a set of clauses S

Resolution System

Background

- Literal $p \quad \overline{p}$
- Clause $p \vee \overline{q} \vee r \vee \ldots \rightarrow p\overline{q}r \ldots$ Empty clause $\perp$
- Input formula $(p \vee q) \wedge (r \vee \overline{p}) \ldots \rightarrow \{pq, r\overline{p}\}$
- Resolution rule $$\frac{pC \qquad \overline{p}D}{CD}\, p$$

  Antecedents: $pC\ \overline{p}D$ Resolvent: CD Pivot: p
- Resolution proof of unsatisfiability of a set of clauses S
  - Tree
  - Leaves as clauses of S
  - Intermediate nodes as resolvents
  - Root as unique empty clause

Resolution Proofs

SAT

- $A \equiv \{\overline{p}\overline{q}, p\overline{q}\}$ $\qquad$ $B \equiv \{q\overline{r}, qr\}$

Resolution Proofs

SAT

- $A \equiv \{\overline{p}\,\overline{q}, p\overline{q}\}$ $\qquad$ $B \equiv \{q\overline{r}, qr\}$

- Proof of unsatisfiability

$$\dfrac{\dfrac{\overline{p}\,\overline{q} \qquad p\overline{q}}{\overline{q}}\;p \qquad \dfrac{q\overline{r} \qquad qr}{q}\;r}{\perp}\;q$$

Interpolant Generation

SAT [Pudlák97]

- Computation of interpolant I for $A \wedge B$ from proof structure

Interpolant Generation

SAT [Pudlák97]

- Computation of interpolant I for $A \wedge B$ from proof structure
- Partial interpolant for leaf

Interpolant Generation

SAT [Pudlák97]

- Computation of interpolant I for $A \wedge B$ from proof structure
- Partial interpolant for leaf
- Partial interpolant for resolvent
  - Pivot
  - Partial interpolants for antecedents

- Computation of interpolant I for $A \wedge B$ from proof structure
- Partial interpolant for leaf
- Partial interpolant for resolvent
  - Pivot
  - Partial interpolants for antecedents
- Partial interpolant for $\perp$ is I

Interpolant Generation

SAT [Pudlák97]

- $A \equiv \{\overline{p}\,\overline{q}, p\overline{q}\}$ $\qquad$ $B \equiv \{q\overline{r}, qr\}$

- Proof of unsatisfiability

$$\dfrac{\dfrac{\overline{p}\,\overline{q} \qquad p\overline{q}}{\overline{q}}\;p \qquad\qquad \dfrac{q\overline{r} \qquad qr}{q}\;r}{\bot}\;q$$

- $A \equiv \{\overline{p}\,\overline{q}, p\overline{q}\}$ $\qquad$ $B \equiv \{q\overline{r}, qr\}$

- Proof of unsatisfiability

$$\dfrac{\dfrac{\overline{p}\,\overline{q}\ \ \{\bot\} \qquad p\overline{q}\ \ \{\bot\}}{\overline{q}}\,p \qquad\qquad \dfrac{q\overline{r} \qquad qr}{q}\,r}{\bot}\,q$$

Interpolant Generation

SAT [Pudlák97]

- $A \equiv \{\overline{p}\overline{q}, p\overline{q}\} \qquad B \equiv \{q\overline{r}, qr\}$
- Proof of unsatisfiability

$$\dfrac{\dfrac{\overline{p}\overline{q}\ \{\bot\} \qquad p\overline{q}\ \{\bot\}}{\overline{q}}\,p \qquad \dfrac{q\overline{r}\ \{\top\} \qquad qr\ \{\top\}}{q}\,r}{\bot}\,q$$

Interpolant Generation

SAT [Pudlák97]

- $A \equiv \{\overline{p}\overline{q}, p\overline{q}\}$ $\qquad$ $B \equiv \{q\overline{r}, qr\}$
- Proof of unsatisfiability

$$\dfrac{\dfrac{\overline{p}\overline{q}\ \ \{\bot\} \qquad\qquad p\overline{q}\ \ \{\bot\}}{\overline{q}\ \ \{\bot \vee \bot\}}\,p \qquad\qquad \dfrac{q\overline{r}\ \ \{\top\} \qquad\qquad qr\ \ \{\top\}}{q}\,r}{\bot}\,q$$

Interpolant Generation

SAT [Pudlák97]

- $A \equiv \{\overline{p}\,\overline{q}, p\overline{q}\}$ $\qquad$ $B \equiv \{q\overline{r}, qr\}$
- Proof of unsatisfiability

$$\dfrac{\dfrac{\overline{p}\,\overline{q}\ \ \{\bot\} \qquad p\overline{q}\ \ \{\bot\}}{\overline{q}\ \ \{\bot\}}\,p \qquad \dfrac{q\overline{r}\ \ \{\top\} \qquad qr\ \ \{\top\}}{q}\,r}{\bot}\,q$$

Interpolant Generation

SAT [Pudlák97]

- $A \equiv \{\overline{p}\,\overline{q}, p\overline{q}\}$ $\qquad$ $B \equiv \{q\overline{r}, qr\}$

- Proof of unsatisfiability

$$\dfrac{\dfrac{\overline{p}\,\overline{q}\ \ \{\bot\} \qquad p\overline{q}\ \ \{\bot\}}{\overline{q}\ \ \{\bot\}}\,p \qquad \dfrac{q\overline{r}\ \ \{\top\} \qquad qr\ \ \{\top\}}{q\ \ \{\top \wedge \top\}}\,r}{\bot}\,q$$

Interpolant Generation

SAT [Pudlák97]

- $A \equiv \{\overline{p}\,\overline{q}, p\overline{q}\}$ $\qquad B \equiv \{q\overline{r}, qr\}$

- Proof of unsatisfiability

$$\dfrac{\dfrac{\overline{p}\,\overline{q}\ \ \{\bot\} \qquad p\overline{q}\ \ \{\bot\}}{\overline{q}\ \ \{\bot\}}\,p \qquad \dfrac{q\overline{r}\ \ \{\top\} \qquad qr\ \ \{\top\}}{q\ \ \{\top\}}\,r}{\bot}\,q$$

- $A \equiv \{\overline{p}\overline{q}, p\overline{q}\}$ $\qquad B \equiv \{q\overline{r}, qr\}$

- Proof of unsatisfiability

$$
\dfrac{\dfrac{\overline{p}\overline{q}\ \ \{\bot\} \qquad p\overline{q}\ \ \{\bot\}}{\overline{q}\ \ \{\bot\}}\,p \qquad \dfrac{q\overline{r}\ \ \{\top\} \qquad qr\ \ \{\top\}}{q\ \ \{\top\}}\,r}{\bot\ \ \{(\bot \vee \overline{q}) \wedge (\top \vee q)\}}\,q
$$

Interpolant Generation

SAT [Pudlák97]

- $A \equiv \{\overline{p}\overline{q}, p\overline{q}\}$ $\qquad$ $B \equiv \{q\overline{r}, qr\}$
- Proof of unsatisfiability

$$\dfrac{\dfrac{\overline{p}\overline{q}\ \ \{\bot\} \qquad p\overline{q}\ \ \{\bot\}}{\overline{q}\ \ \{\bot\}}\,p \qquad \dfrac{q\overline{r}\ \ \{\top\} \qquad qr\ \ \{\top\}}{q\ \ \{\top\}}\,r}{\bot\ \ \{\overline{q}\}}\,q$$

Resolution Proofs

SMT

- $A \equiv \{\overbrace{\boxed{(5x - y \leq 1)}}^{p}, \overbrace{\boxed{(y - 5x \leq -1)}}^{q}\}, B \equiv \{\overbrace{\boxed{(y - 5z \leq 3)}}^{r}, \overbrace{\boxed{(5z - y \leq \; -2)}}^{s}\}$

Resolution Proofs

SMT

- $A \equiv \{\overbrace{\boxed{(5x - y \leq 1)}}^{p}, \overbrace{\boxed{(y - 5x \leq -1)}}^{q}\}, B \equiv \{\overbrace{\boxed{(y - 5z \leq 3)}}^{r}, \overbrace{\boxed{(5z - y \leq \ -2)}}^{s}\}$

- Theory lemmata

Resolution Proofs

SMT

- $A \equiv \{\overbrace{\boxed{(5x - y \leq 1)}}^{p}, \overbrace{\boxed{(y - 5x \leq -1)}}^{q}\}, B \equiv \{\overbrace{\boxed{(y - 5z \leq 3)}}^{r}, \overbrace{\boxed{(5z - y \leq\ -2)}}^{s}\}$
- Theory lemmata
  - $\mathcal{LIA}$: $\overbrace{\boxed{(x - z \leq 0)}}^{t}\overbrace{\boxed{(x - z \geq 1)}}^{u}$

Resolution Proofs

SMT

- $A \equiv \{\overbrace{\boxed{(5x - y \leq 1)}}^{p}, \overbrace{\boxed{(y - 5x \leq -1)}}^{q}\}, B \equiv \{\overbrace{\boxed{(y - 5z \leq 3)}}^{r}, \overbrace{\boxed{(5z - y \leq -2)}}^{s}\}$
- Theory lemmata
  - $\mathcal{LIA}$: $\overbrace{\boxed{(x - z \leq 0)}}^{t}\overbrace{\boxed{(x - z \geq 1)}}^{u}$
  - $\mathcal{LRA}$: $\overbrace{\boxed{(5x - y \not\leq 1)}}^{\overline{p}}\overbrace{\boxed{(y - 5z \not\leq 3)}}^{\overline{r}}\overbrace{\boxed{(x - z \not\geq 1)}}^{\overline{u}}$

Resolution Proofs

SMT

- $A \equiv \{\overbrace{\boxed{(5x - y \leq 1)}}^{p}, \overbrace{\boxed{(y - 5x \leq -1)}}^{q}\}, B \equiv \{\overbrace{\boxed{(y - 5z \leq 3)}}^{r}, \overbrace{\boxed{(5z - y \leq -2)}}^{s}\}$
- Theory lemmata
  - $\mathcal{LIA}$: $\overbrace{\boxed{(x - z \leq 0)}}^{t} \overbrace{\boxed{(x - z \geq 1)}}^{u}$
  - $\mathcal{LRA}$: $\overbrace{\boxed{(5x - y \not\leq 1)}}^{\overline{p}} \overbrace{\boxed{(y - 5z \not\leq 3)}}^{\overline{r}} \overbrace{\boxed{(x - z \not\geq 1)}}^{\overline{u}}$
  - $\mathcal{LRA}$: $\overbrace{\boxed{(y - 5x \not\leq -1)}}^{\overline{q}} \overbrace{\boxed{(5z - y \not\leq -2)}}^{\overline{s}} \overbrace{\boxed{(x - z \not\leq 0)}}^{\overline{t}}$

Resolution Proofs

SMT

- $A \equiv \{p, q\}$ $\qquad B \equiv \{r, s\}$ $\qquad L \equiv \{tu, \overline{pr}\,\overline{u}, \overline{qs}\,\overline{t}\}$

- $A \equiv \{p, q\} \qquad B \equiv \{r, s\} \qquad L \equiv \{tu, \overline{p}\,\overline{r}\,\overline{u}, \overline{q}\,\overline{s}\,\overline{t}\}$

- Proof of unsatisfiability

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{p \qquad \overline{p}\,\overline{r}\,\overline{u}}{\overline{r}\,\overline{u}}\;p \qquad r}{\overline{u}}\;r \qquad tu}{t}\;u \qquad \overline{q}\,\overline{s}\,\overline{t}}{\overline{q}\,\overline{s}}\;t \qquad q}{\overline{s}}\;q \qquad s}{\perp}\;s$$

- $A \equiv \{p, q\} \qquad B \equiv \{r, s\} \qquad L \equiv \{tu, \overline{p}\,\overline{r}\,\overline{u}, \overline{q}\,\overline{s}\,\overline{t}\}$

- Proof of unsatisfiability

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{p \qquad \overline{p}\,\overline{r}\,\overline{u}}{\overline{r}\,\overline{u}}\;p \qquad r}{\overline{u}}\;r \qquad tu}{t}\;u \qquad \overline{q}\,\overline{s}\,\overline{t}}{\overline{q}\,\overline{s}}\;t \qquad q}{\overline{s}}\;q \qquad s}{\bot}\;s$$

Interpolant Generation

SMT

- $A \equiv \{p, q\}$ $\qquad B \equiv \{r, s\}$ $\qquad L \equiv \{tu, \overline{p}\,\overline{r}\,\overline{u}, \overline{q}\,\overline{s}\,\overline{t}\}$

- Proof of unsatisfiability

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{p\ \ \{\bot\} \qquad \overline{p}\,\overline{r}\,\overline{u}}{\overline{r}\,\overline{u}}\ p \qquad r}{\overline{u}}\ r \qquad tu}{t}\ u \qquad \overline{q}\,\overline{s}\,\overline{t}}{\overline{q}\,\overline{s}}\ t \qquad q\ \ \{\bot\}}{\overline{s}}\ q \qquad s}{\bot}\ s$$

Interpolant Generation
SMT

- $A \equiv \{p, q\} \qquad B \equiv \{r, s\} \qquad L \equiv \{tu, \overline{p}\,\overline{r}\,\overline{u}, \overline{q}\,\overline{s}\,\overline{t}\}$
- Proof of unsatisfiability

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{p\ \ \{\bot\} \qquad \overline{p}\,\overline{r}\,\overline{u}}{\overline{r}\,\overline{u}}\;p \qquad r\ \ \{\top\}}{\overline{u}}\;r \qquad tu}{t}\;u \qquad \overline{q}\,\overline{s}\,\overline{t}}{\overline{q}\,\overline{s}}\;t \qquad q\ \ \{\bot\}}{\overline{s}}\;q \qquad s\ \ \{\top\}}{\bot}\;s
$$

Interpolant Generation

SMT

- $A \equiv \{p, q\} \qquad B \equiv \{r, s\} \qquad L \equiv \{tu, \overline{p}\,\overline{r}\,\overline{u}, \overline{q}\,\overline{s}\,\overline{t}\}$

- Proof of unsatisfiability

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{p\ \{\bot\} \qquad \overline{p}\,\overline{r}\,\overline{u}}{\overline{r}\,\overline{u}}\,p \qquad r\ \{\top\}}{\overline{u}}\,r \qquad tu\ \ ?}{t}\,u \qquad \overline{q}\,\overline{s}\,\overline{t}}{\overline{q}\,\overline{s}}\,t \qquad q\ \{\bot\}}{\overline{s}}\,q \qquad s\ \{\top\}}{\bot}\,s$$

Interpolation

Challenge

- State-of-the-art approach [Pudlák97, McMillan04]

Interpolation
Challenge

- State-of-the-art approach [Pudlák97, McMillan04]
  - Derivation of unsatisfiability proof of $A \wedge B$
  - Computation of interpolant from proof structure in linear time

Interpolation

Challenge

- State-of-the-art approach [Pudlák97, McMillan04]
  - Derivation of unsatisfiability proof of $A \wedge B$
  - Computation of interpolant from proof structure in linear time
- Restriction

Interpolation

Challenge

- State-of-the-art approach [Pudlák97, McMillan04]
  - Derivation of unsatisfiability proof of $A \wedge B$
  - Computation of interpolant from proof structure in linear time
- Restriction
  - Need for proof not to contain AB-mixed predicates

A-local B-local AB-common AB-mixed

Interpolation
Challenge

- State-of-the-art approach [Pudlák97, McMillan04]
  - Derivation of unsatisfiability proof of $A \wedge B$
  - Computation of interpolant from proof structure in linear time
- Restriction
  - Need for proof not to contain AB-mixed predicates

A-local B-local AB-common AB-mixed

$A \equiv \{ \boxed{(5x - y \leq 1)}, \ldots \}$ $B \equiv \{ \boxed{(y - 5z \leq 3)}, \ldots \}$

- State-of-the-art approach [Pudlák97, McMillan04]
  - Derivation of unsatisfiability proof of $A \wedge B$
  - Computation of interpolant from proof structure in linear time
- Restriction
  - Need for proof not to contain AB-mixed predicates

A-local B-local AB-common AB-mixed

$A \equiv \{ \boxed{(5x - y \leq 1)}, \ldots \}$ $B \equiv \{ \boxed{(y - 5z \leq 3)}, \ldots \}$

$L \equiv \{ \boxed{(x - z \leq 0)}, \ldots \}$

- Need for proof not to contain AB-mixed predicates

- Need for proof not to contain AB-mixed predicates
- Tune solvers to avoid generating AB-mixed predicates [Cimatti08,Beyer08]

- Need for proof not to contain AB-mixed predicates
- Tune solvers to avoid generating AB-mixed predicates [Cimatti08,Beyer08]
- Transform proof to remove AB-mixed predicates

- Proof transformation approach

- Proof transformation approach
- Motivation: more flexibility by decoupling SMT solving and interpolant generation

Proof Transformation

Motivation

- Proof transformation approach
- Motivation: more flexibility by decoupling SMT solving and interpolant generation
- Motivation: standard SMT techniques can require addition of AB-mixed predicates

- Proof transformation approach
- Motivation: more flexibility by decoupling SMT solving and interpolant generation
- Motivation: standard SMT techniques can require addition of AB-mixed predicates
  - Theory reduction via Lemma on Demand [DeMoura02, Barrett06]

    Reduction of $\mathcal{AX}$ to $\mathcal{EUF}$

    Reduction of $\mathcal{LIA}$ to $\mathcal{LRA}$

    Ackermann's Expansion
  - Theory combination via DTC [Bozzano05]

Proof Transformation Framework

- Proof rewriting framework based on local rules

Proof Transformation Framework

- Proof rewriting framework based on local rules
- Isolation of AB-mixed predicates into subtrees

Proof Transformation Framework

- Proof rewriting framework based on local rules
- Isolation of AB-mixed predicates into subtrees
- Removal of AB-mixed subtrees

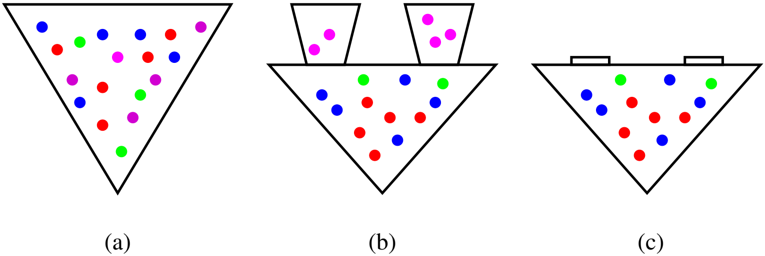
Proof Transformation Framework

- Proof rewriting framework based on local rules
- Isolation of AB-mixed predicates into subtrees
- Removal of AB-mixed subtrees
- No more AB-mixed predicates, proof still valid

Proof Transformation

Effect

(a) Initial proof: A-local, B-local, AB-common, AB-mixed
(b) Transformed proof: AB-mixed predicates isolated into subtrees
(c) Final proof: AB-mixed subtrees removed, new leaves are theory lemmata



Proof Transformation

Advantages

- No more AB-mixed predicates, new leaves are theory lemmata

Proof Transformation

Advantages

- No more AB-mixed predicates, new leaves are theory lemmata
- Easy combination of SMT and interpolation techniques

Proof Transformation

Advantages

- No more AB-mixed predicates, new leaves are theory lemmata
- Easy combination of SMT and interpolation techniques
  - Theory reduction, theory combination without restrictions

Proof Transformation

Advantages

- No more AB-mixed predicates, new leaves are theory lemmata
- Easy combination of SMT and interpolation techniques
  - Theory reduction, theory combination without restrictions
  - Interpolant generation for propositional resolution proofs of unsatisfiability [Pudlák97]

Proof Transformation

Advantages

- No more AB-mixed predicates, new leaves are theory lemmata
- Easy combination of SMT and interpolation techniques
  - Theory reduction, theory combination without restrictions
  - Interpolant generation for propositional resolution proofs of unsatisfiability [Pudlák97]
  - (Partial) interpolant generation for theory (combination) lemmata [Yorsh05]

Proof Transformation Framework

Features

- Local rewriting rules

- Local rewriting rules

- Rule context

$$\dfrac{\dfrac{pqC \qquad \overline{p}D}{qCD}\,p \qquad \overline{q}E}{CDE}\,q$$

Proof Transformation Framework

Features

- Local rewriting rules

- Rule context

$$\dfrac{\dfrac{pqC \qquad \overline{p}D}{qCD}\,p \qquad \overline{q}E}{CDE}\,q$$

- Exhaustiveness up to symmetry

Proof Transformation Framework

Local Rewriting Rules

- $$\dfrac{\dfrac{pqC \qquad \overline{p}D}{qCD}\,p \qquad \overline{q}E}{CDE}\,q \quad \Rightarrow \quad \dfrac{\dfrac{pqC \qquad \overline{q}E}{pCE}\,q \qquad \overline{p}D}{CDE}\,p$$

Proof Transformation Framework

Local Rewriting Rules

- $$\dfrac{\dfrac{pqC \qquad \overline{p}D}{qCD}\,p \qquad \overline{q}E}{CDE}\,q \quad \Rightarrow \quad \dfrac{\dfrac{pqC \qquad \overline{q}E}{pCE}\,q \qquad \overline{p}D}{CDE}\,p$$

- Pivots swapping

Proof Transformation Framework

Local Rewriting Rules

- $$\dfrac{\dfrac{pqC \qquad \overline{p}D}{qCD}\,p \qquad \overline{q}E}{CDE}\,q \quad \Rightarrow \quad \dfrac{\dfrac{pqC \qquad \overline{q}E}{pCE}\,q \qquad \overline{p}D}{CDE}\,p$$

- Pivots swapping

- AB-mixed predicates isolation into subtrees

Reduction $\mathcal{LIA}$ to $\mathcal{LRA}$

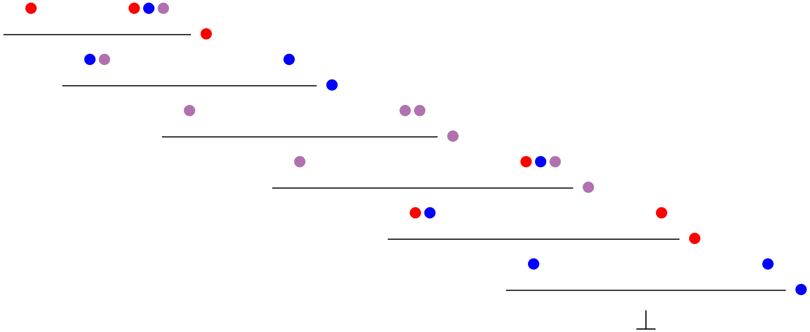
Transformation

- $A \equiv \{p, q\} \qquad B \equiv \{r, s\} \qquad L \equiv \{tu, \overline{p}\,\overline{r}\,\overline{u}, \overline{q}\,\overline{s}\,\overline{t}\}$

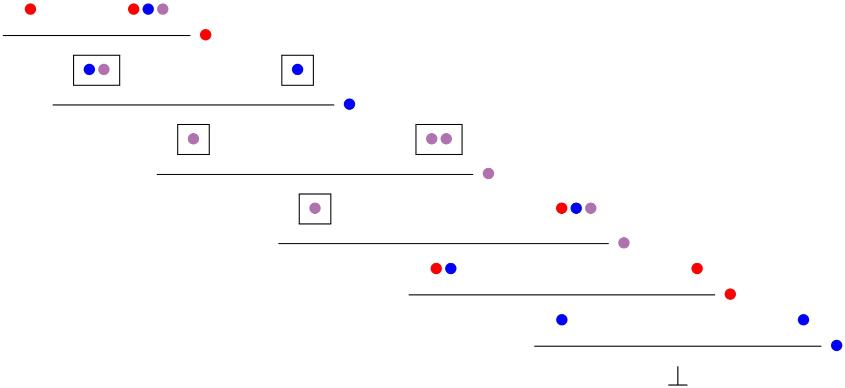
- Proof of unsatisfiability

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{p \qquad \overline{p}\,\overline{r}\,\overline{u}}{\overline{r}\,\overline{u}}\,p \qquad r}{\overline{u}}\,r \qquad tu}{t}\,u \qquad \overline{q}\,\overline{s}\,\overline{t}}{\overline{q}\,\overline{s}}\,t \qquad q}{\overline{s}}\,q \qquad s}{\bot}\,s$$

- Proof of unsatisfiability



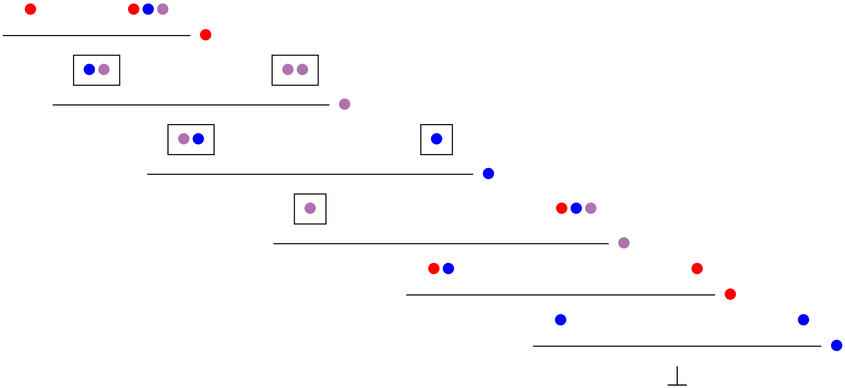
- Proof of unsatisfiability



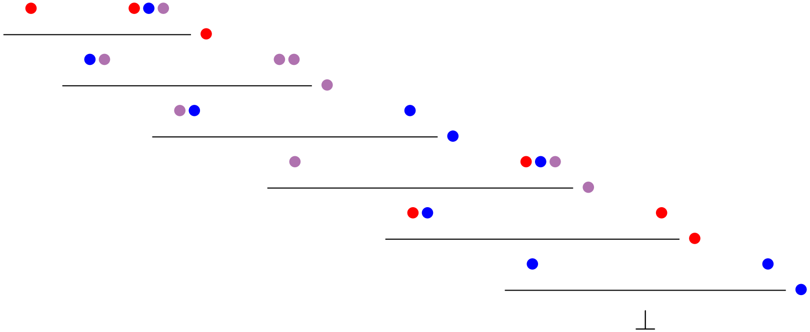
Reduction $\mathcal{LIA}$ to $\mathcal{LRA}$

Transformation

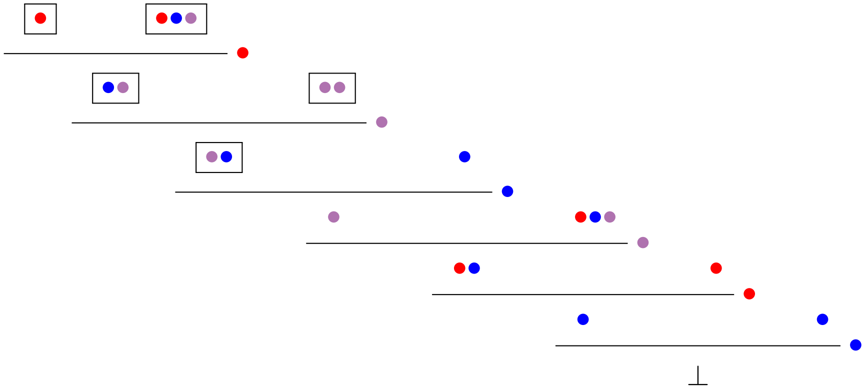
- Proof of unsatisfiability



- Proof of unsatisfiability



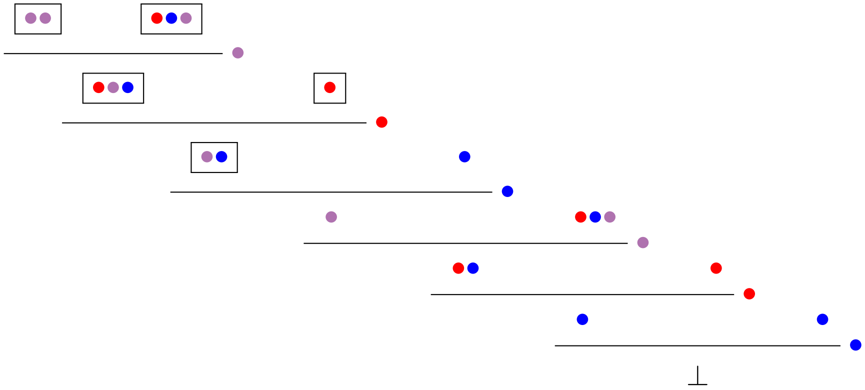
- Proof of unsatisfiability



Reduction $\mathcal{LIA}$ to $\mathcal{LRA}$

Transformation

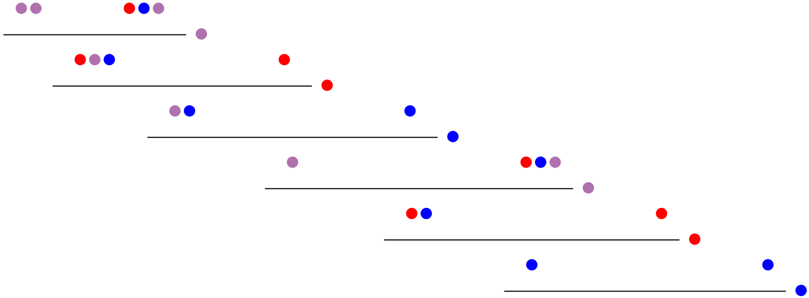
- Proof of unsatisfiability



Reduction $\mathcal{LIA}$ to $\mathcal{LRA}$

Transformation

- Proof of unsatisfiability

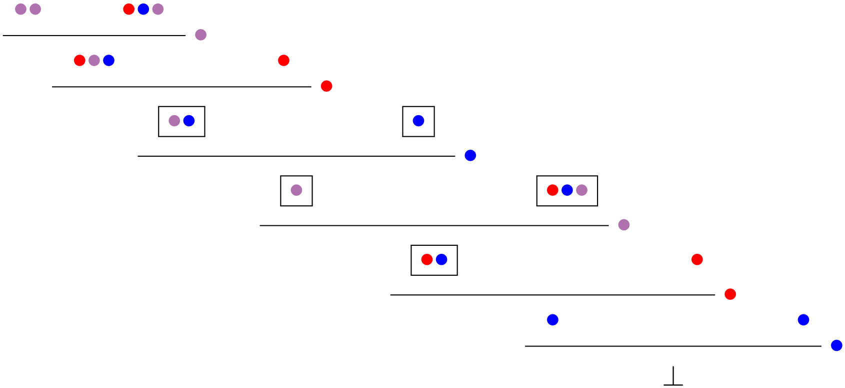


$\perp$

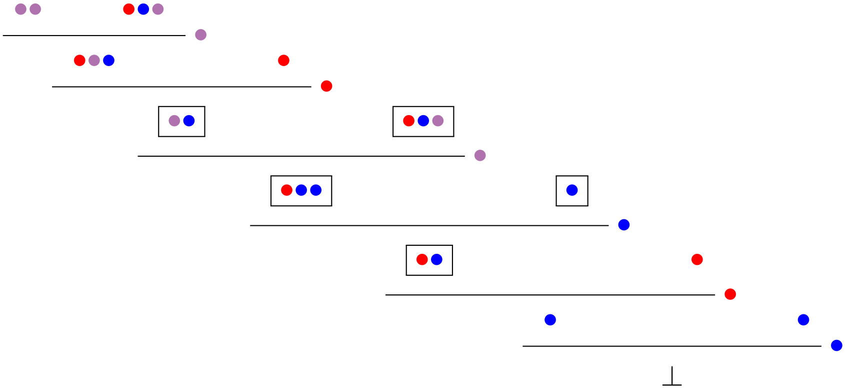
Reduction $\mathcal{LIA}$ to $\mathcal{LRA}$

Transformation

- Proof of unsatisfiability



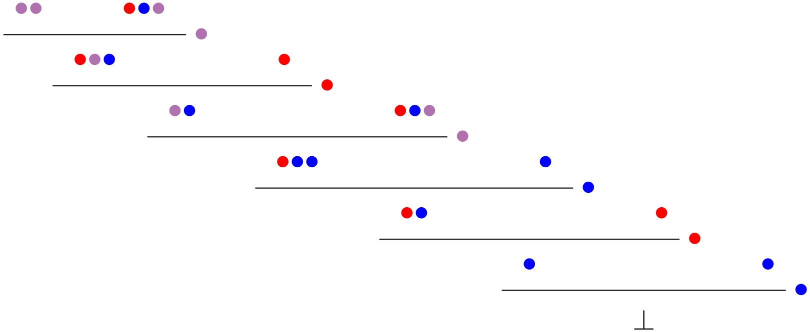
- Proof of unsatisfiability



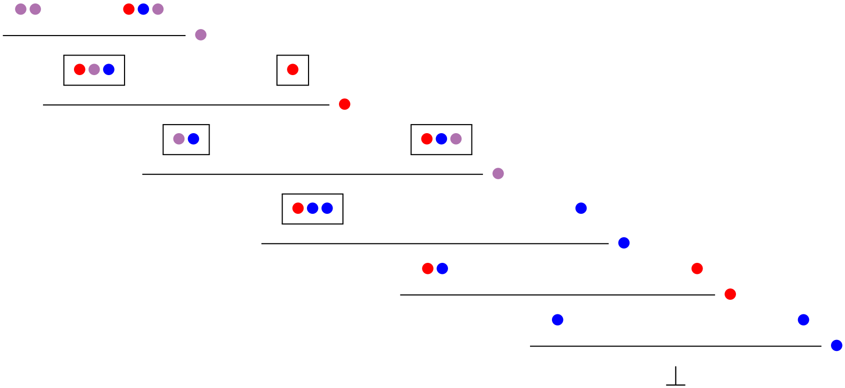
Reduction $\mathcal{LIA}$ to $\mathcal{LRA}$

Transformation

- Proof of unsatisfiability



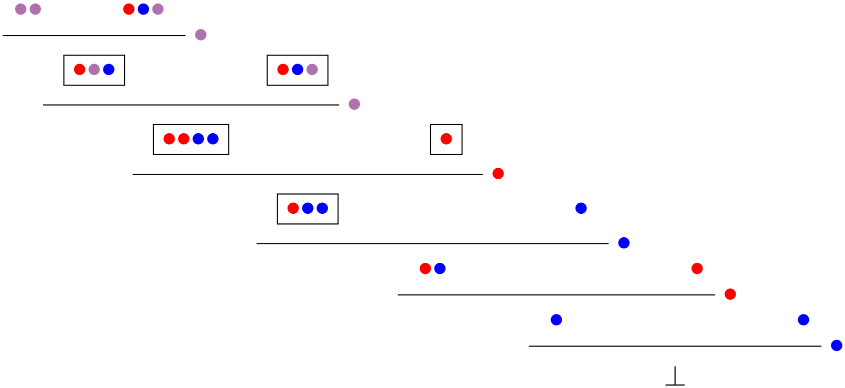
- Proof of unsatisfiability



Reduction $\mathcal{LIA}$ to $\mathcal{LRA}$

Transformation

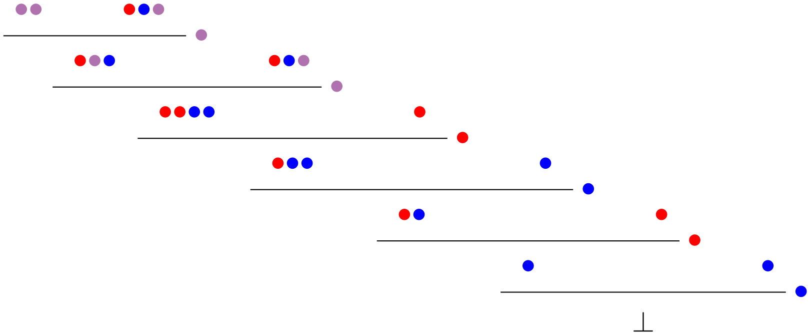
- Proof of unsatisfiability



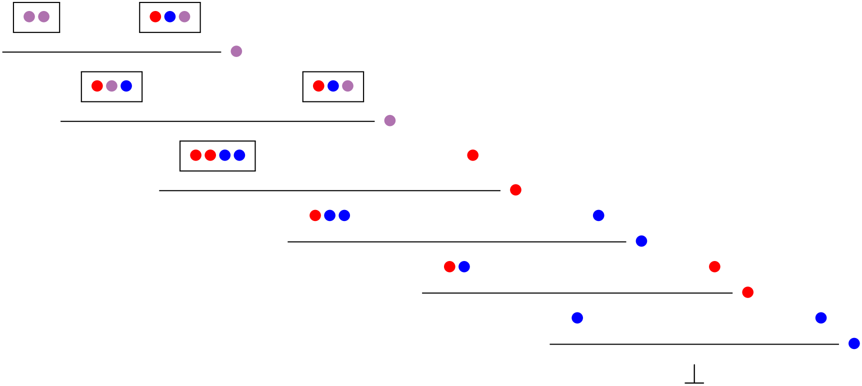
Reduction $\mathcal{LIA}$ to $\mathcal{LRA}$

Transformation

- Proof of unsatisfiability



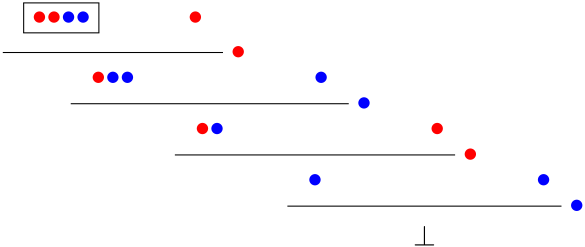
- Proof of unsatisfiability



Reduction $\mathcal{LIA}$ to $\mathcal{LRA}$

Transformation

- Proof of unsatisfiability



Proof Transformation Framework

Considerations

- Potential drawbacks

Proof Transformation Framework

Considerations

- Potential drawbacks
  - Overhead w.r.t. solving time

- Potential drawbacks
  - Overhead w.r.t. solving time
  - Increase of proof size

- Local rewriting rules

Transformation Framework

Features

- Local rewriting rules
  - B reduction
  - A perturbation

Transformation Framework

Features

- Local rewriting rules
  - B reduction
  - A perturbation

- Rule context

$$\dfrac{\dfrac{pqC \qquad \overline{p}D}{qCD}\,p \qquad \overline{q}E}{CDE}\,q$$

Transformation Framework

Features

- Local rewriting rules
  - B reduction
  - A perturbation
- Rule context

$$\dfrac{\dfrac{pqC \qquad \overline{p}D}{qCD}\,p \qquad \overline{q}E}{CDE}\,q$$

- Exhaustiveness up to symmetry

Transformation Framework

Local rewriting rules

- B rules

$B1$	$\dfrac{\dfrac{pqC \qquad \overline{p}qD}{qCD}\,p \qquad p\overline{q}E}{pCDE}\,q \quad \Rightarrow \quad \dfrac{pqC \qquad p\overline{q}E}{pCE}\,q$

Transformation Framework

Local rewriting rules

- B rules

$B1$	$\dfrac{\dfrac{pqC \qquad \overline{p}qD}{qCD}\,p \qquad\qquad p\overline{q}E}{pCDE}\,q \quad \Rightarrow \quad \dfrac{pqC \qquad p\overline{q}E}{pCE}\,q$

- Redundancy as reintroduction variable after elimination

Transformation Framework

Local rewriting rules

- B rules

$B1$	$\dfrac{\dfrac{pqC \qquad \overline{p}qD}{qCD}\,p \qquad p\overline{q}E}{pCDE}\,q \quad \Rightarrow \quad \dfrac{pqC \qquad p\overline{q}E}{pCE}\,q$

  - Redundancy as reintroduction variable after elimination
  - Subproof simplification

Transformation Framework

Local rewriting rules

- B rules

$B1$	$$\dfrac{\dfrac{pqC \qquad \bar{p}qD}{qCD}\,p \qquad p\bar{q}E}{pCDE}\,q$$	$\Rightarrow$	$$\dfrac{pqC \qquad p\bar{q}E}{pCE}\,q$$

  - Redundancy as reintroduction variable after elimination
  - Subproof simplification
  - Subproof root strengthening

Transformation Framework

Local rewriting rules

- A rules

$A2$	$\dfrac{\dfrac{pqC \qquad \overline{p}D}{qCD}\,p \qquad \overline{q}E}{CDE}\,q \quad \Rightarrow \quad \dfrac{\dfrac{pqC \qquad \overline{q}E}{pCE}\,q \qquad \overline{p}D}{CDE}\,p$

Transformation Framework

Local rewriting rules

- A rules

$A2$	$\dfrac{\dfrac{pqC \qquad \overline{p}D}{qCD}\,p \qquad \overline{q}E}{CDE}\,q \quad \Rightarrow \quad \dfrac{\dfrac{pqC \qquad \overline{q}E}{pCE}\,q \qquad \overline{p}D}{CDE}\,p$

  - Pivots swapping

Transformation Framework

Local rewriting rules

- A rules

$A2$	$$\dfrac{\dfrac{pqC \qquad \overline{p}D}{qCD}\,p \qquad \overline{q}E}{CDE}\,q \quad \Rightarrow \quad \dfrac{\dfrac{pqC \qquad \overline{q}E}{pCE}\,q \qquad \overline{p}D}{CDE}\,p$$

  - Pivots swapping

  - Topology perturbation

Transformation Framework

Local rewriting rules

- A rules

$A2$	$\dfrac{\dfrac{pqC \qquad \overline{p}D}{qCD}\,p \qquad \overline{q}E}{CDE}\,q \quad \Rightarrow \quad \dfrac{\dfrac{pqC \qquad \overline{q}E}{pCE}\,q \qquad \overline{p}D}{CDE}\,p$

  - Pivots swapping
  - Topology perturbation
  - Redundancies exposure

Local rewriting rules

$A1$	$\dfrac{\dfrac{pqC \qquad \overline{p}qD}{qCD}\,p \qquad \overline{q}E}{CDE}\,q$	$\Rightarrow$	$\dfrac{\dfrac{pqC \qquad \overline{q}E}{pCE} \qquad \dfrac{\overline{q}E \qquad \overline{p}qD}{\overline{p}DE}\,q}{CDE}\,p$
$A2$	$\dfrac{\dfrac{pqC \qquad \overline{p}D}{qCD}\,p \qquad \overline{q}E}{CDE}\,q$	$\Rightarrow$	$\dfrac{\dfrac{pqC \qquad \overline{q}E}{pCE}\,q \qquad \overline{p}D}{CDE}\,p$
$B1$	$\dfrac{\dfrac{pqC \qquad \overline{p}qD}{qCD}\,p \qquad p\overline{q}E}{pCDE}\,q$	$\Rightarrow$	$\dfrac{pqC \qquad p\overline{q}E}{pCE}\,q$
$B2$	$\dfrac{\dfrac{pqC \qquad \overline{p}D}{qDC}\,p \qquad p\overline{q}E}{pCDE}\,q$	$\Rightarrow$	$\dfrac{\dfrac{pqC \qquad p\overline{q}E}{pCE}\,q \qquad \overline{p}D}{CDE}\,p$
$B2'$	$\dfrac{\dfrac{pqC \qquad \overline{p}D}{qDC}\,p \qquad p\overline{q}E}{pCDE}\,q$	$\Rightarrow$	$\dfrac{pqC \qquad p\overline{q}E}{pCE}\,q$
$B3$	$\dfrac{\dfrac{pqC \qquad \overline{p}D}{qCD}\,p \qquad \overline{p}\overline{q}E}{\overline{p}CDE}\,q$	$\Rightarrow$	$\overline{p}D$

Evaluation

Framework and Benchmarks

- OpenSMT

Evaluation

Framework and Benchmarks

- OpenSMT
  - C++ open-source SMT solver developed at USI
  - Fastest open-source solver in SMT-comp 2009, 2010 for various logics

Evaluation

Framework and Benchmarks

- OpenSMT
  - C++ open-source SMT solver developed at USI
  - Fastest open-source solver in SMT-comp 2009, 2010 for various logics
- Benchmarks

Evaluation

Framework and Benchmarks

- OpenSMT
  - C++ open-source SMT solver developed at USI
  - Fastest open-source solver in SMT-comp 2009, 2010 for various logics
- Benchmarks
  - SMT: SMT-LIB library
  - Academic and industrial problems

Experimental results over QF_UFIDL

Group	#	$\#AB$	$\%_{time}$	$\%_{nodes}$	$\%_{edges}$
RDS	2	7	84%	-16%	-19%
EufLaAr	2	74	18%	187%	193%
pete	15	20	16%	66%	68%
pete2	52	13	6%	73%	80%
uclid	11	12	29%	87%	90%
Overall	82	16	13%	74%	79%

- # — number of benchmarks solved
- $\#AB$ — average number of AB-mixed predicates in proof
- $\%_{time}$ — average time overhead
- $\%_{nodes}$, $\%_{edges}$ — average difference in proof size

Comparison

- RecyclePivots (closest related work) [Strichman'08]

Comparison

- RecyclePivots (closest related work) [Strichman'08]
  - **Pros**
    Global information
    Fast and effective
  - **Cons**
    Cannot expose redundancies

Comparison

- RecyclePivots (closest related work) [Strichman'08]
  - **Pros**
    Global information
    Fast and effective
  - **Cons**
    Cannot expose redundancies
- Rule-based approach

Comparison

- RecyclePivots (closest related work) [Strichman'08]
  - **Pros**
    Global information
    Fast and effective
  - **Cons**
    Cannot expose redundancies

- Rule-based approach
  - **Pros**
    Flexibility in rules application
    Flexibility in amount of transformation
    Can expose redundancies
  - **Cons**
    Local information

Implementation
Reduction Algorithm

- Based on a sequence of proof traversals (e.g. topological order)

Implementation

Reduction Algorithm

- Based on a sequence of proof traversals (e.g. topological order)
- Parameterized in number of traversals and time limit

Implementation

Reduction Algorithm

- Based on a sequence of proof traversals (e.g. topological order)
- Parameterized in number of traversals and time limit
- Examination non-leaf clauses

Implementation

Reduction Algorithm

- Based on a sequence of proof traversals (e.g. topological order)
- Parameterized in number of traversals and time limit
- Examination non-leaf clauses
  - Pivot in both antecedents $\rightarrow$ update, match context, apply rule

$$\dfrac{qC'D' \qquad \overline{q}E'}{CDE}\,q \quad \Rightarrow \quad \dfrac{qC'D' \qquad \overline{q}E'}{C'D'E'}\,q \quad \Rightarrow \quad \dfrac{\dfrac{pqC' \qquad \overline{p}D'}{qC'D'}\,p \qquad \overline{q}E'}{C'D'E'}\,q$$

Implementation

Reduction Algorithm

- Based on a sequence of proof traversals (e.g. topological order)
- Parameterized in number of traversals and time limit
- Examination non-leaf clauses
  - Pivot in both antecedents → update, match context, apply rule

$$\frac{qC'D' \qquad \overline{q}E'}{CDE}\,q \quad \Rightarrow \quad \frac{qC'D' \qquad \overline{q}E'}{C'D'E'}\,q \quad \Rightarrow \quad \cfrac{\cfrac{pqC' \qquad \overline{p}D'}{qC'D'}\,p \qquad \overline{q}E'}{C'D'E'}\,q$$

  - Pivot not in both antecedents → remove resolution step

$$\frac{C'D' \qquad \overline{q}E'}{CDE}\,q \quad \Rightarrow \quad C'D'$$

Implementation

Reduction Algorithm

- Based on a sequence of proof traversals (e.g. topological order)
- Parameterized in number of traversals and time limit
- Examination non-leaf clauses
  - Pivot in both antecedents → update, match context, apply rule

$$\dfrac{qC'D' \qquad \overline{q}E'}{CDE}\,q \quad \Rightarrow \quad \dfrac{qC'D' \qquad \overline{q}E'}{C'D'E'}\,q \quad \Rightarrow \quad \dfrac{\dfrac{pqC' \qquad \overline{p}D'}{qC'D'}\,p \qquad \overline{q}E'}{C'D'E'}\,q$$

  - Pivot not in both antecedents → remove resolution step

$$\dfrac{C'D' \qquad \overline{q}E'}{CDE}\,q \quad \Rightarrow \quad C'D'$$

- Easy combination with RecyclePivots

Evaluation

Framework and Benchmarks

- Implemented in C++ and integrated with OpenSMT
- Available at **www.inf.usi.ch/phd/rollini/hvc.html**

Evaluation

Framework and Benchmarks

- Implemented in C++ and integrated with OpenSMT
- Available at **www.inf.usi.ch/phd/rollini/hvc.html**

- Benchmarks

Evaluation

Framework and Benchmarks

- Implemented in C++ and integrated with OpenSMT
- Available at **www.inf.usi.ch/phd/rollini/hvc.html**
- Benchmarks
  - SMT: SMT-LIB library
  - SAT: SAT competition
  - Academic and industrial problems

Combined Approach Evaluation

Experimental results over SMT: QF_UF, QF_IDL, QF_LRA, QF_RDL

	#	Avg_{nodes}	Avg_{edges}	Avg_{core}	$T(s)$	Max_{nodes}	Max_{edges}	Max_{core}
RP	1370	6.7%	7.5%	1.3%	1.7	65.1%	68.9%	39.1%
Ratio								
0.01	1366	8.9%	10.7%	1.4%	3.4	66.3%	70.2%	45.7%
0.025	1366	9.8%	11.9%	1.5%	3.6	77.2%	79.9%	45.7%
0.05	1366	10.7%	13.0%	1.6%	4.1	78.5%	81.2%	45.7%
0.075	1366	11.4%	13.8%	1.7%	4.5	78.5%	81.2%	45.7%
0.1	1364	11.8%	14.4%	1.7%	5.0	78.8%	83.6%	45.7%
0.25	1359	13.6%	16.6%	1.9%	7.6	79.6%	84.4%	45.7%
0.5	1348	15.0%	18.4%	2.0%	11.5	79.1%	85.2%	45.7%
0.75	1341	16.0%	19.5%	2.1%	15.1	79.9%	86.1%	45.7%
1	1337	16.7%	20.4%	2.2%	18.8	79.9%	86.1%	45.7%

- $Ratio$ — time threshold as fraction of solving time
- # — number of benchmarks solved
- Avg_{nodes}, Avg_{edges}, Avg_{core} — average reduction in proof size
- $T(s)$ — average transformation time in seconds
- Max_{nodes}, Max_{edges}, Max_{core} — max reduction in proof size

Combined Approach Evaluation

Experimental results over SMT: QF_UF, QF_IDL, QF_LRA, QF_RDL

	#	Avg_{nodes}	Avg_{edges}	Avg_{core}	$T(s)$	Max_{nodes}	Max_{edges}	Max_{core}
RP	1370	6.7%	7.5%	1.3%	1.7	65.1%	68.9%	39.1%
Ratio								
0.01	1366	8.9%	10.7%	1.4%	3.4	66.3%	70.2%	45.7%
0.025	1366	9.8%	11.9%	1.5%	3.6	77.2%	79.9%	45.7%
0.05	1366	10.7%	13.0%	1.6%	4.1	78.5%	81.2%	45.7%
0.075	1366	11.4%	13.8%	1.7%	4.5	78.5%	81.2%	45.7%
0.1	1364	11.8%	14.4%	1.7%	5.0	78.8%	83.6%	45.7%
0.25	1359	13.6%	16.6%	1.9%	7.6	79.6%	84.4%	45.7%
0.5	1348	15.0%	18.4%	2.0%	11.5	79.1%	85.2%	45.7%
0.75	1341	16.0%	19.5%	2.1%	15.1	79.9%	86.1%	45.7%
1	1337	16.7%	20.4%	2.2%	18.8	79.9%	86.1%	45.7%

- $Ratio$ — time threshold as fraction of solving time
- # — number of benchmarks solved
- Avg_{nodes}, Avg_{edges}, Avg_{core} — average reduction in proof size
- $T(s)$ — average transformation time in seconds
- Max_{nodes}, Max_{edges}, Max_{core} — max reduction in proof size

Combined Approach Evaluation

Experimental results over SMT: QF_UF, QF_IDL, QF_LRA, QF_RDL

	#	Avg_{nodes}	Avg_{edges}	Avg_{core}	$T(s)$	Max_{nodes}	Max_{edges}	Max_{core}
RP	1370	6.7%	7.5%	1.3%	1.7	65.1%	68.9%	39.1%
Ratio								
0.01	1366	8.9%	10.7%	1.4%	3.4	66.3%	70.2%	45.7%
0.025	1366	9.8%	11.9%	1.5%	3.6	77.2%	79.9%	45.7%
0.05	1366	10.7%	13.0%	1.6%	4.1	78.5%	81.2%	45.7%
0.075	1366	11.4%	13.8%	1.7%	4.5	78.5%	81.2%	45.7%
0.1	1364	11.8%	14.4%	1.7%	5.0	78.8%	83.6%	45.7%
0.25	1359	13.6%	16.6%	1.9%	7.6	79.6%	84.4%	45.7%
0.5	1348	15.0%	18.4%	2.0%	11.5	79.1%	85.2%	45.7%
0.75	1341	16.0%	19.5%	2.1%	15.1	79.9%	86.1%	45.7%
1	1337	16.7%	20.4%	2.2%	18.8	79.9%	86.1%	45.7%

- *Ratio* — time threshold as fraction of solving time
- # — number of benchmarks solved
- Avg_{nodes}, Avg_{edges}, Avg_{core} — average reduction in proof size
- $T(s)$ — average transformation time in seconds
- Max_{nodes}, Max_{edges}, Max_{core} — max reduction in proof size

Combined Approach Evaluation

Experimental results over SAT

	#	Avg_{nodes}	Avg_{edges}	Avg_{core}	$T(s)$	Max_{nodes}	Max_{edges}	Max_{core}
RP	25	5.9%	6.5%	1.7%	10.8	33.1%	33.4%	30.3%
Ratio								
0.01	25	6.8%	7.9%	1.7%	32.3	34.0%	34.4%	30.5%
0.025	25	6.8%	7.9%	1.7%	32.3	34.0%	34.4%	30.5%
0.05	25	7.0%	8.2%	1.8%	40.0	34.0%	34.4%	30.5%
0.075	25	7.2%	8.4%	1.8%	49.3	34.7%	35.1%	30.5%
0.1	25	7.3%	8.4%	1.8%	60.2	34.7%	35.1%	30.5%
0.25	25	7.6%	8.8%	1.9%	125.3	39.8%	40.6%	31.7%
0.5	25	7.8%	9.1%	1.9%	243.5	41.0%	41.9%	32.1%
0.75	25	7.9%	9.3%	1.9%	360.0	41.6%	42.6%	32.1%
1	23	8.4%	9.9%	2.1%	175.6	33.1%	33.4%	30.6%

- $Ratio$ — time threshold as fraction of solving time
- # — number of benchmarks solved
- Avg_{nodes}, Avg_{edges}, Avg_{core} — average reduction in proof size
- $T(s)$ — average transformation time in seconds
- Max_{nodes}, Max_{edges}, Max_{core} — max reduction in proof size

Combined Approach Evaluation

Experimental results over SAT

	#	Avg_{nodes}	Avg_{edges}	Avg_{core}	$T(s)$	Max_{nodes}	Max_{edges}	Max_{core}
RP	25	5.9%	6.5%	1.7%	10.8	33.1%	33.4%	30.3%
Ratio								
0.01	25	6.8%	7.9%	1.7%	32.3	34.0%	34.4%	30.5%
0.025	25	6.8%	7.9%	1.7%	32.3	34.0%	34.4%	30.5%
0.05	25	7.0%	8.2%	1.8%	40.0	34.0%	34.4%	30.5%
0.075	25	7.2%	8.4%	1.8%	49.3	34.7%	35.1%	30.5%
0.1	25	7.3%	8.4%	1.8%	60.2	34.7%	35.1%	30.5%
0.25	25	7.6%	8.8%	1.9%	125.3	39.8%	40.6%	31.7%
0.5	25	7.8%	9.1%	1.9%	243.5	41.0%	41.9%	32.1%
0.75	25	7.9%	9.3%	1.9%	360.0	41.6%	42.6%	32.1%

- $Ratio$ — time threshold as fraction of solving time
- # — number of benchmarks solved
- Avg_{nodes}, Avg_{edges}, Avg_{core} — average reduction in proof size
- $T(s)$ — average transformation time in seconds
- Max_{nodes}, Max_{edges}, Max_{core} — max reduction in proof size

Combined Approach Evaluation

Experimental results over SAT

	#	Avg_{nodes}	Avg_{edges}	Avg_{core}	$T(s)$	Max_{nodes}	Max_{edges}	Max_{core}
RP	25	5.9%	6.5%	1.7%	10.8	33.1%	33.4%	30.3%
Ratio								
0.01	25	6.8%	7.9%	1.7%	32.3	34.0%	34.4%	30.5%
0.025	25	6.8%	7.9%	1.7%	32.3	34.0%	34.4%	30.5%
0.05	25	7.0%	8.2%	1.8%	40.0	34.0%	34.4%	30.5%
0.075	25	7.2%	8.4%	1.8%	49.3	34.7%	35.1%	30.5%
0.1	25	7.3%	8.4%	1.8%	60.2	34.7%	35.1%	30.5%
0.25	25	7.6%	8.8%	1.9%	125.3	39.8%	40.6%	31.7%
0.5	25	7.8%	9.1%	1.9%	243.5	41.0%	41.9%	32.1%
0.75	25	7.9%	9.3%	1.9%	360.0	41.6%	42.6%	32.1%

- *Ratio* — time threshold as fraction of solving time
- # — number of benchmarks solved
- Avg_{nodes}, Avg_{edges}, Avg_{core} — average reduction in proof size
- $T(s)$ — average transformation time in seconds
- Max_{nodes}, Max_{edges}, Max_{core} — max reduction in proof size

Conclusion

- OpenSMT Solver
- Application to Lazy Abstraction with Interpolants
- Proof Manipulation for Interpolation and Reduction
- `http://verify.inf.usi.ch`

Thanks

References

R. Bruttomesso, S. Rollini, N. Sharygina, and A. Tsitovich.
Flexible Interpolation with Local Proof Transformations.
In ICCAD, 2010.

R. Bruttomesso, E. Pek, N. Sharygina, and A. Tsitovich.
The OpenSMT Solver.
In TACAS, pages 150153, 2010.

S. Rollini, R. Bruttomesso, and N. Sharygina.
An Efficient and Flexible Approach to Resolution Proof Reduction.
In HVC, 2010.

References

[Pudlák97] P. Pudlák. *Lower Bounds for Resolution and Cutting Plane Proofs and Monotone Computations*. J. Symb. Log. 1997.

[McMillan04] K. L. McMillan. *An Interpolating Theorem Prover*. TACAS. 2004.

[Cimatti08] A. Cimatti, A. Griggio, R. Sebastiani. *Efficient Interpolant Generation in SMT*. TACAS. 2008.

[Beyer08] D. Beyer, D. Zufferey, R. Majumdar. *CSIsat: Interpolation for LA+EUF*. CAV. 2008.

[Bozzano05] M. Bozzano, R. Bruttomesso, A. Cimatti, T. Junttila, P. Van Rossum, S. Ranise, R. Sebastiani. *Efficient Satisfiability Modulo Theories via Delayed Theory Combination*. CAV. 2005.

[Yorsh05] G. Yorsh, M. Musuvathi. *A Combination Method for Generating Interpolants*. CADE. 2005.