Università della Svizzera italiana

Farkas-Based Tree Interpolation



Sepideh Asadi [1]   Martin Blicha [1]   Antti Hyvärinen [1]


Grigory Fedyukovich [2]   Natasha Sharygina [1]

1- Università della Svizzera italiana (USI), Lugano, Switzerland

2- Florida State University, Tallahassee, USA

SAS 2020, ~~Chicago, USA~~ Online

From Binary to Tree Interpolants

- Lots of binary interpolation algorithms exist for linear arithmetic over reals (LRA)
  - Flexible interpolants (strength, size) important for efficient over-approximation & convergence of program verification
- Applications requiring the tree interpolation property have little choice for LRA interpolation algorithms
  - Limited their scalability or soundness, e.g. incremental verification of program revisions [1], solving non-recursive Horn clauses [2].
- Combining LRA & Propositional interpolation algorithms for over-approximation of bit-vectors in softwares model checking for scalability

[1] Asadi et al, FMCAD 2020 [2] Gupta et al, APLAS 2011

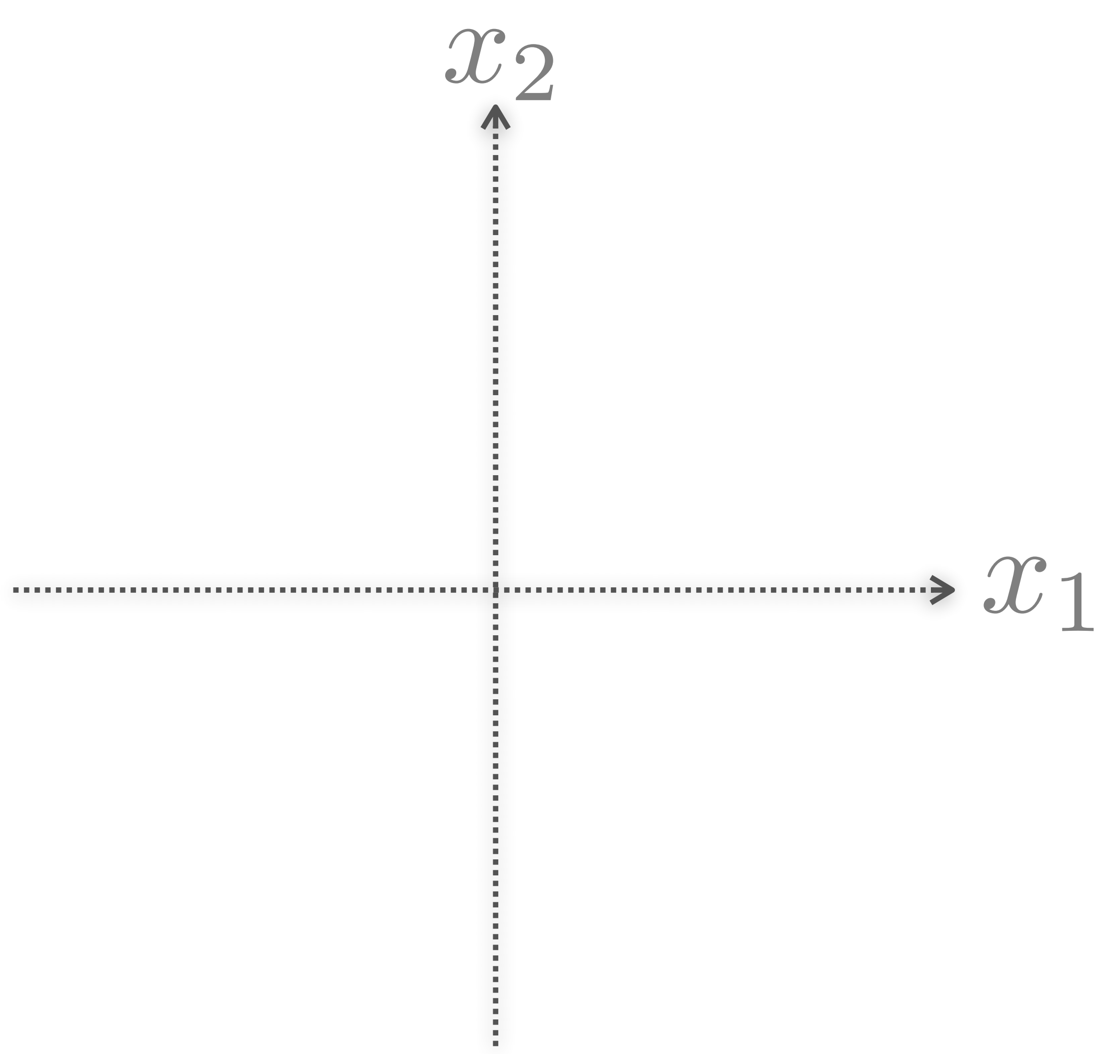
From Binary to Tree Interpolants

- Lots of binary interpolation algorithms exist for linear arithmetic over reals (LRA)
  - Flexible interpolants (strength, size) important for efficient over-approximation & convergence of program verification
- Applications requiring the tree interpolation property have little choice for LRA interpolation algorithms
  - Limited their scalability or soundness, e.g. incremental verification of program revisions [1], solving non-recursive Horn clauses [2].
- Combining LRA & Propositional interpolation algorithms for over-approximation of bit-vectors in softwares model checking for scalability

Goal: Investigate Binary Interpolation algorithms in LRA & Propositional to guarantee Tree Interpolation Property?

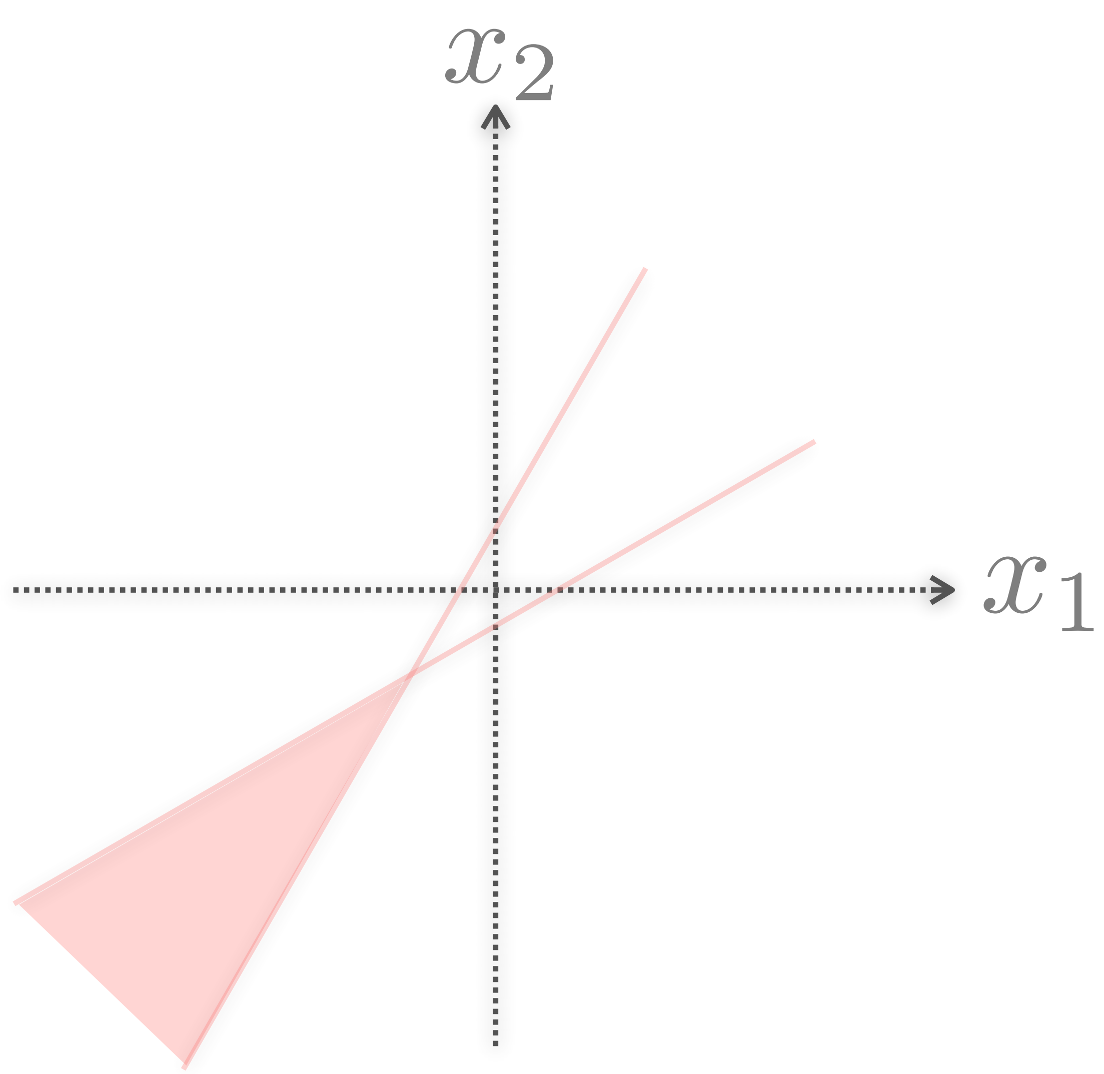
[1] Asadi et al, FMCAD 2020 [2] Gupta et al, APLAS 2011

Example



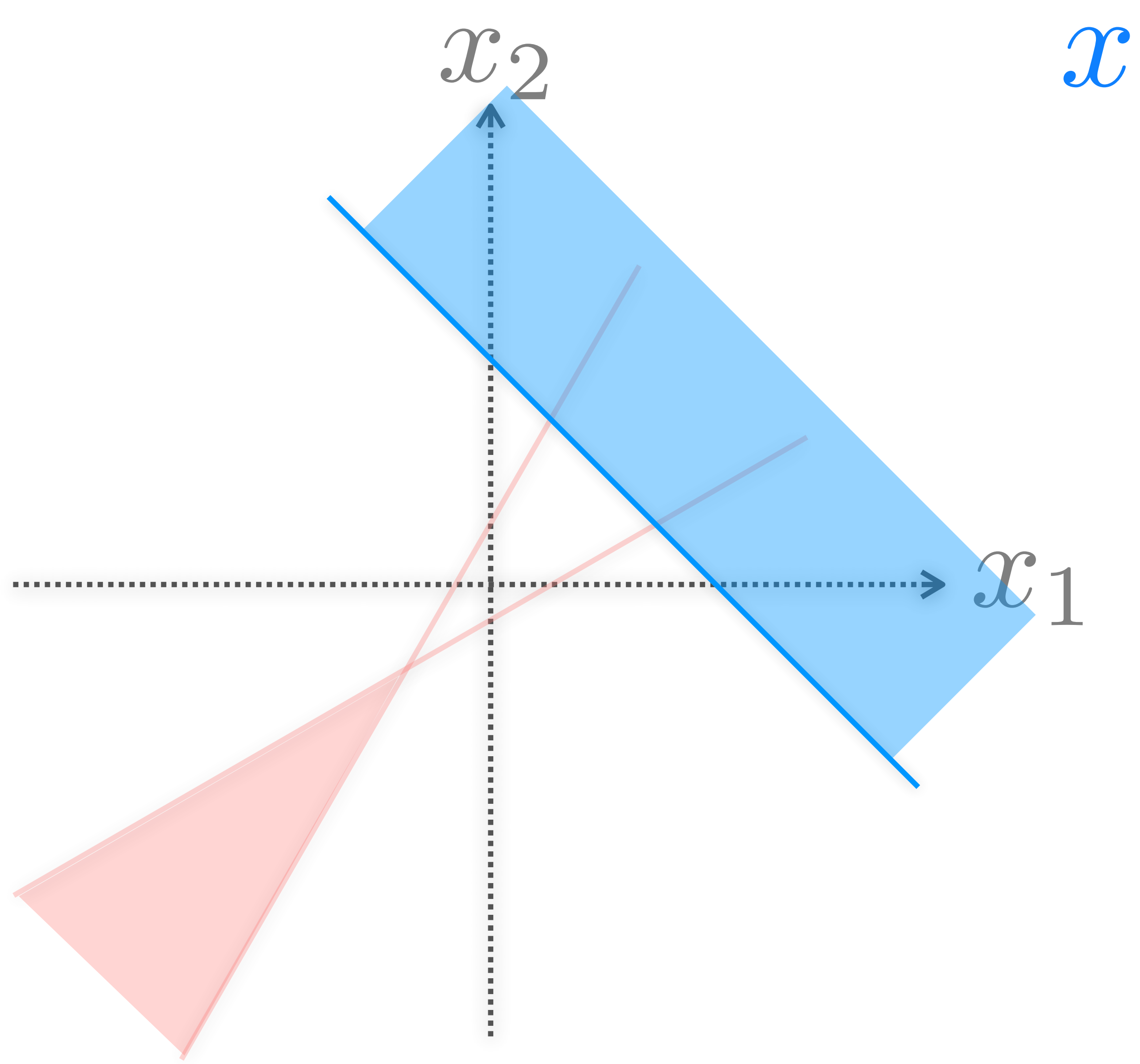
Example

$$-x_1 + 2x_2 \leq -1$$
$$2x_1 - x_2 \leq -1$$



Example

$$-x_1 + 2x_2 \leq -1$$
$$2x_1 - x_2 \leq -1$$



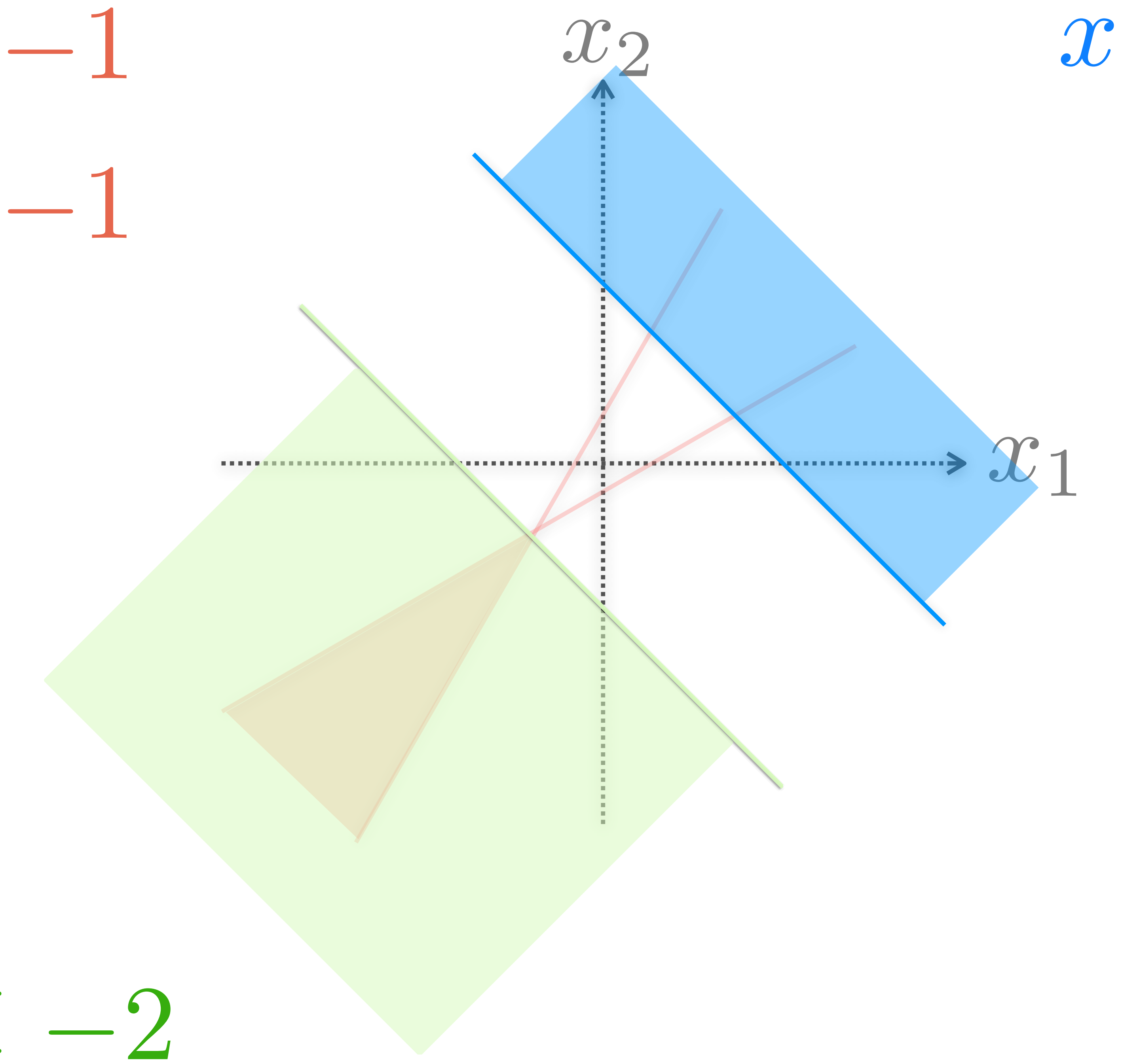
$$x_1 + x_2 \geq 2$$

Example

$$-x_1 + 2x_2 \leq -1$$

$$2x_1 - x_2 \leq -1$$

$$x_1 + x_2 \geq 2$$



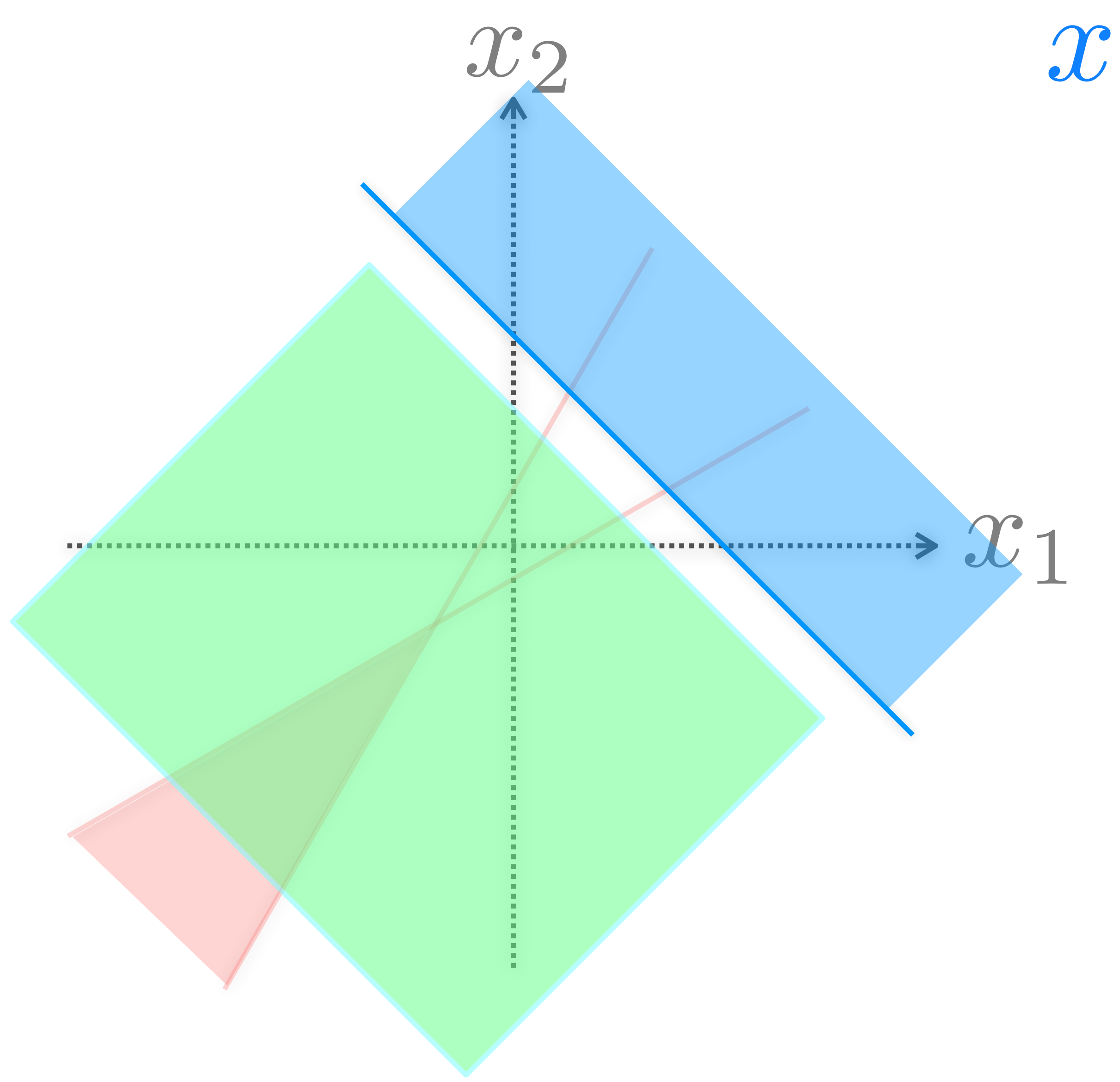
$$x_1 + x_2 \leq -2$$

Example

$$-x_1 + 2x_2 \leq -1$$

$$2x_1 - x_2 \leq -1$$

$$x_1 + x_2 \geq 2$$

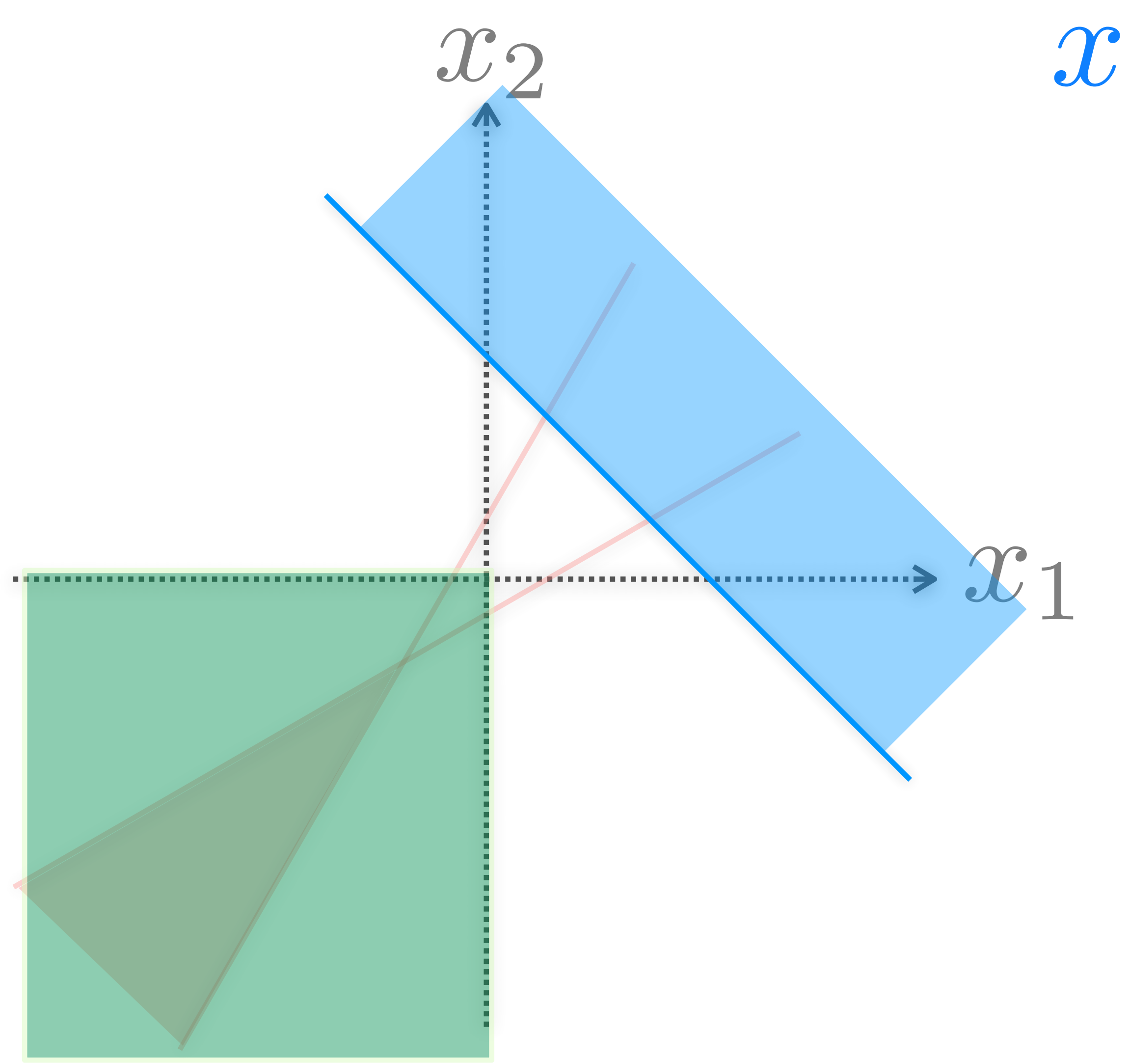


$$x_1 + x_2 \leq 1$$

Example

$$-x_1 + 2x_2 \leq -1$$
$$2x_1 - x_2 \leq -1$$

$$x_1 + x_2 \geq 2$$

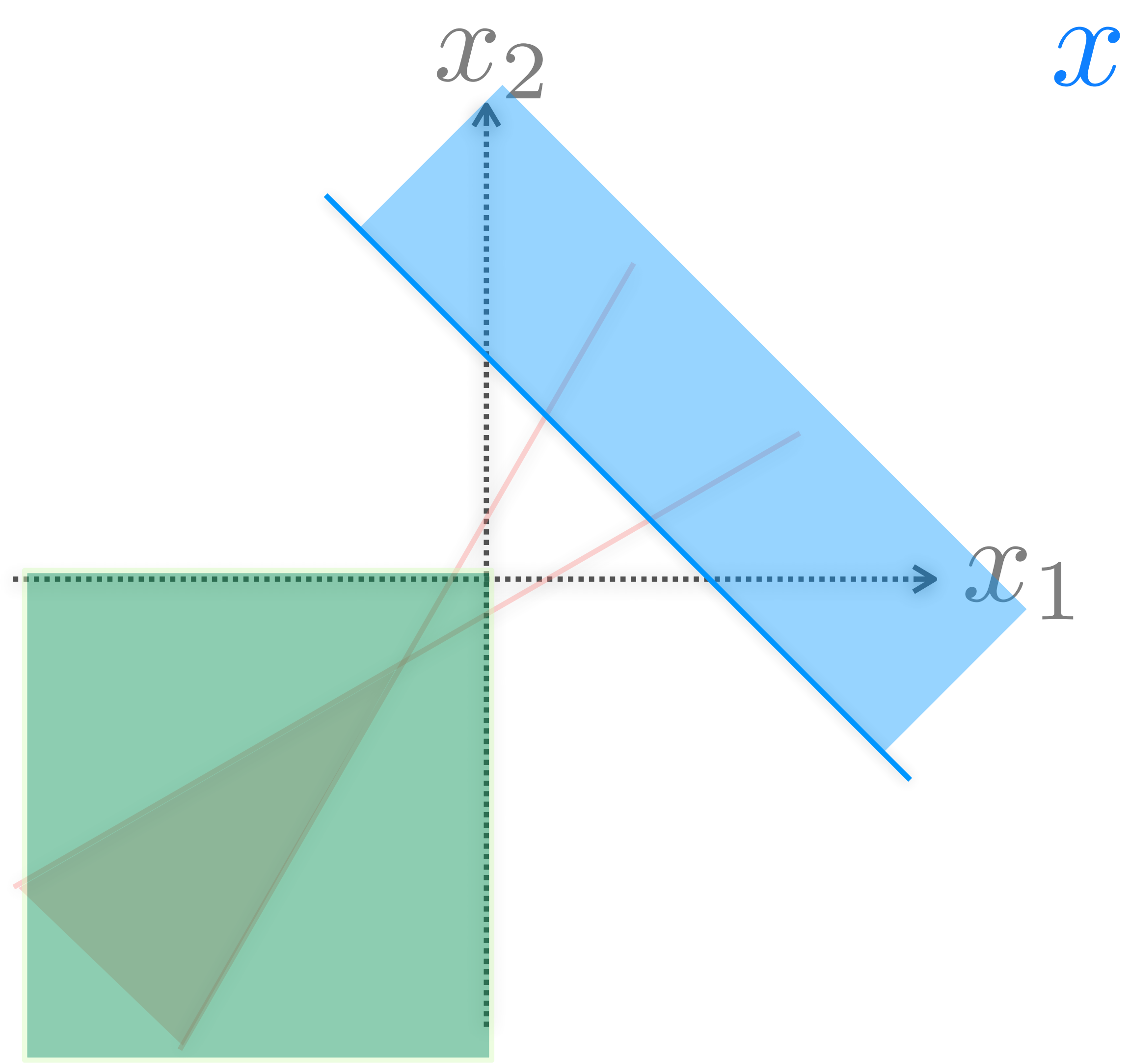


$$x_1 \leq 0$$
$$x_2 \leq 0$$

Example

$$-x_1 + 2x_2 \leq -1$$
$$2x_1 - x_2 \leq -1$$

$$x_1 + x_2 \geq 2$$



$$x_1 \leq 0$$
$$x_2 \leq 0$$

How to compute such over-approximations?

Craig/Binary Interpolants

[Craig'57]

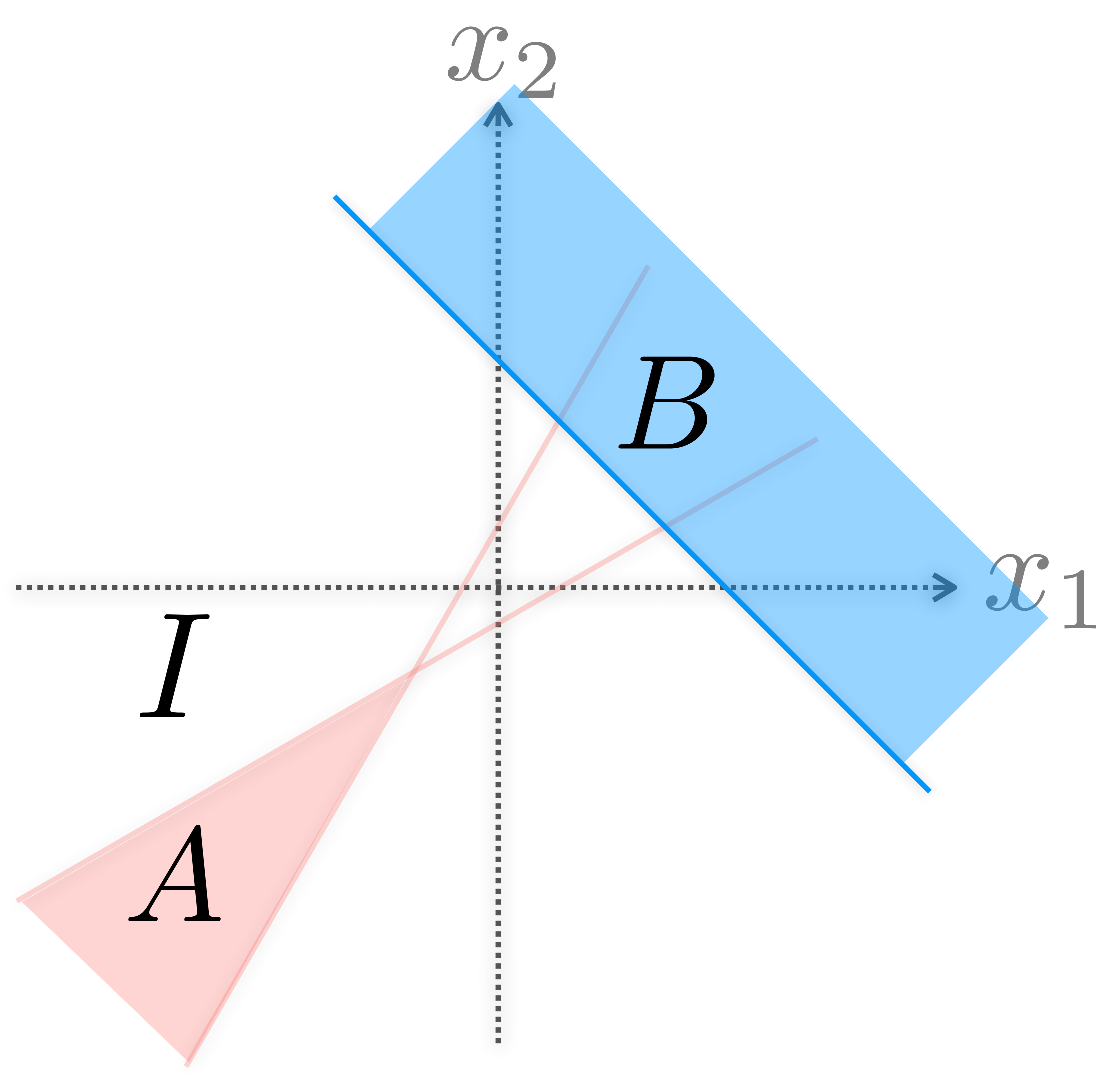
For mutually unsatisfiable A and B,

I is an interpolant for $(A \mid B)$ if:

1) $A \implies I$

2) $I \implies \neg B$

3) I uses only common symbols of A and B



Craig/Binary Interpolants

[Craig'57]

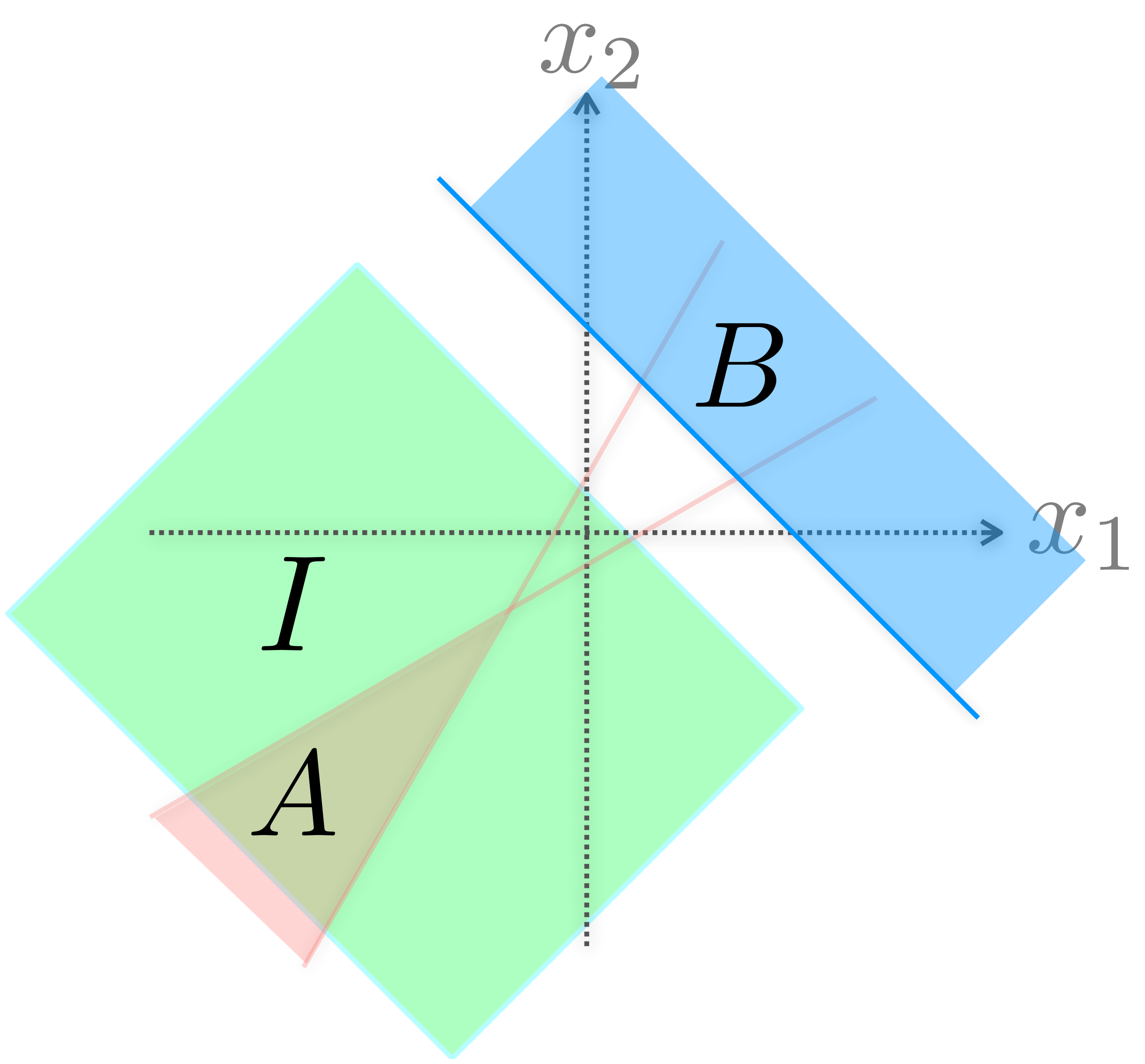
For mutually unsatisfiable A and B,

I is an interpolant for $(A \mid B)$ if:

1) $A \implies I$

2) $I \implies \neg B$

3) I uses only common symbols of A and B



Craig/Binary Interpolants

[Craig'57]

Dual Interpolants

[Alt et al. FMCAD'17]

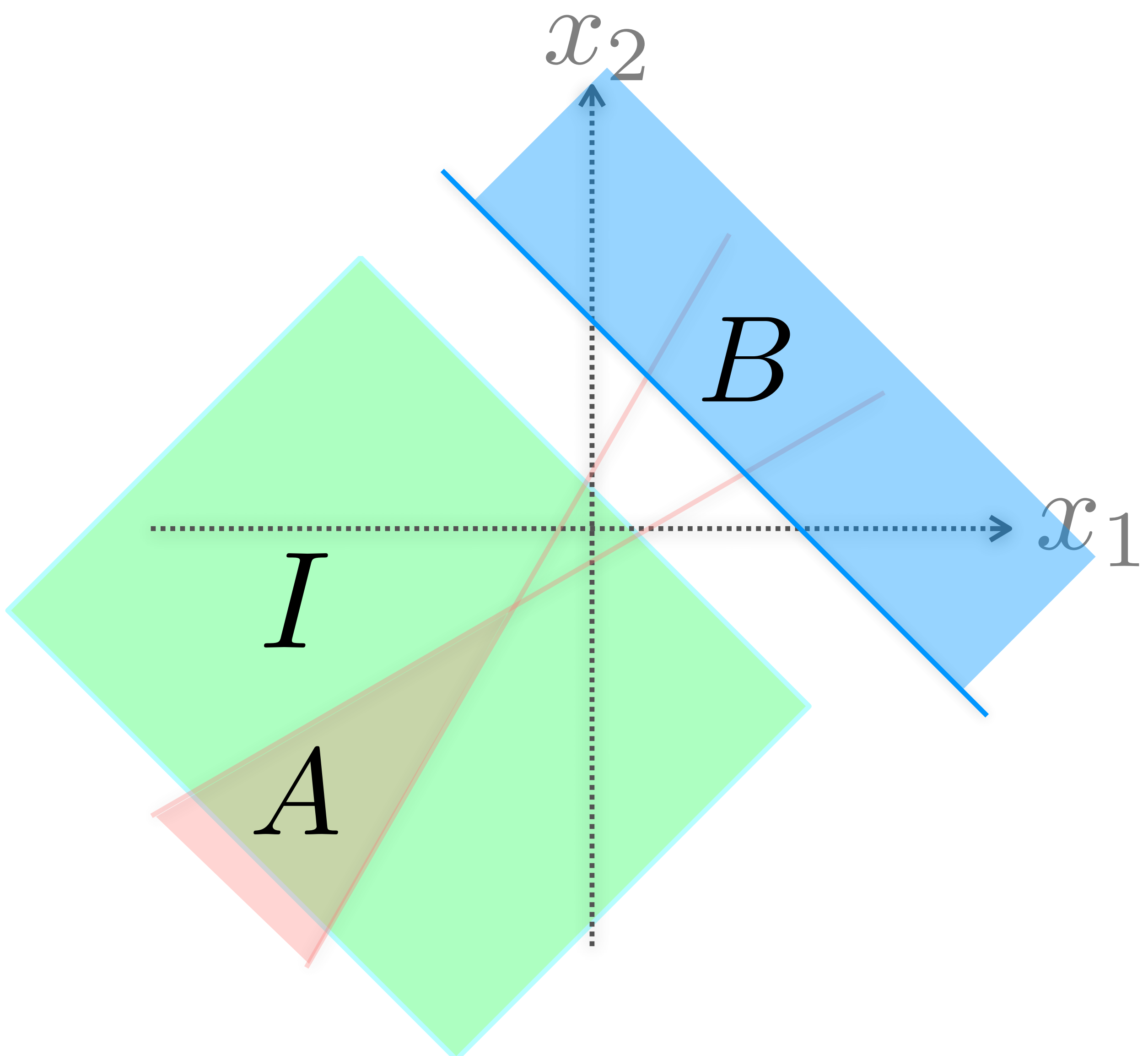
For mutually unsatisfiable A and B,

I is an interpolant for $(A \mid B)$ if:

1) $A \implies I$

2) $I \implies \neg B$

3) I uses only common symbols of A and B



Craig/Binary Interpolants

[Craig'57]

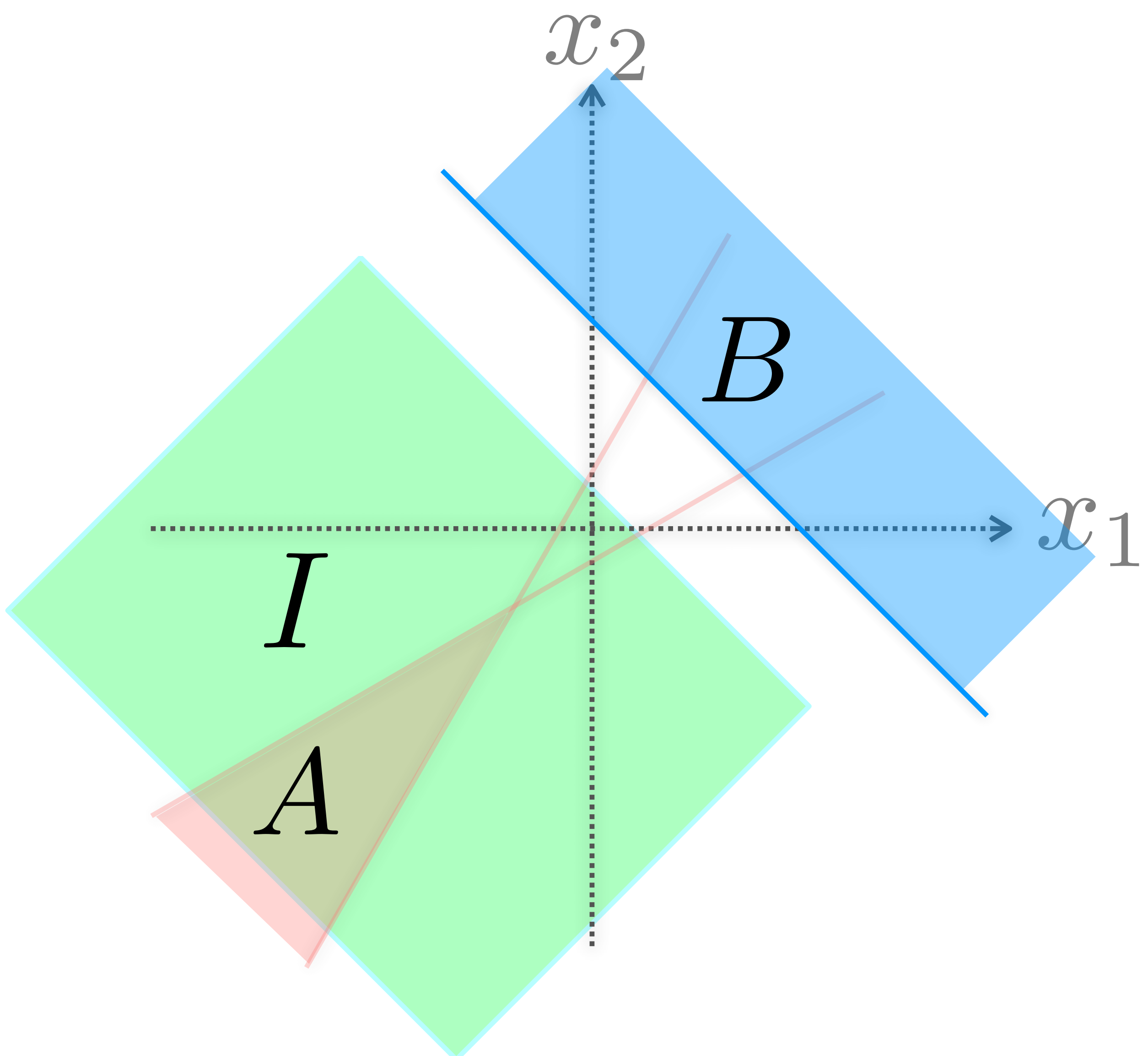
For mutually unsatisfiable A and B,

I is an interpolant for $(A \mid B)$ if:

1) $A \implies I$

2) $I \implies \neg B$

3) I uses only common symbols of A and B



Dual Interpolants

[Alt et al. FMCAD'17]

If I is an interpolant for $(A \mid B)$, then

$\neg I$ is an interpolant for $(B \mid A)$:

1) $B \implies I$

2) $I \implies \neg A$

Tree Interpolation Property (TIP)

TIP relates interpolants computed by multiple binary interpolation problems over the same proof.

Definition: Let $A \land B \land C$ be an unsatisfiable formula and

I_A, I_B, I_{AB} be binary interpolants for interpolation problems:

$$(A \mid B \land C),$$
$$(B \mid A \land C),$$
$$(A \land B \mid C).$$

The tuple $(I_A,\ I_B,\ I_{AB})$ has **strong TIP** iff: $I_A \ \land\ I_B \implies I_{AB}$

The tuple $(I_A,\ I_B,\ I_{AB})$ has **weak TIP** iff: $I_A \ \land\ B \implies I_{AB}$

Tree Interpolation Property (TIP)

TIP relates interpolants computed by multiple binary interpolation problems over the same proof.

Definition: Let $A \wedge B \wedge C$ be an unsatisfiable formula and

I_A, I_B, I_{AB} be binary interpolants for interpolation problems:

$$(A \mid B \wedge C),$$
$$(B \mid A \wedge C),$$
$$(A \wedge B \mid C).$$

This presentation only considers 3 partitions for simplicity!

The tuple $(I_A,\ I_B,\ I_{AB})$ has **strong TIP** iff: $I_A \wedge I_B \implies I_{AB}$

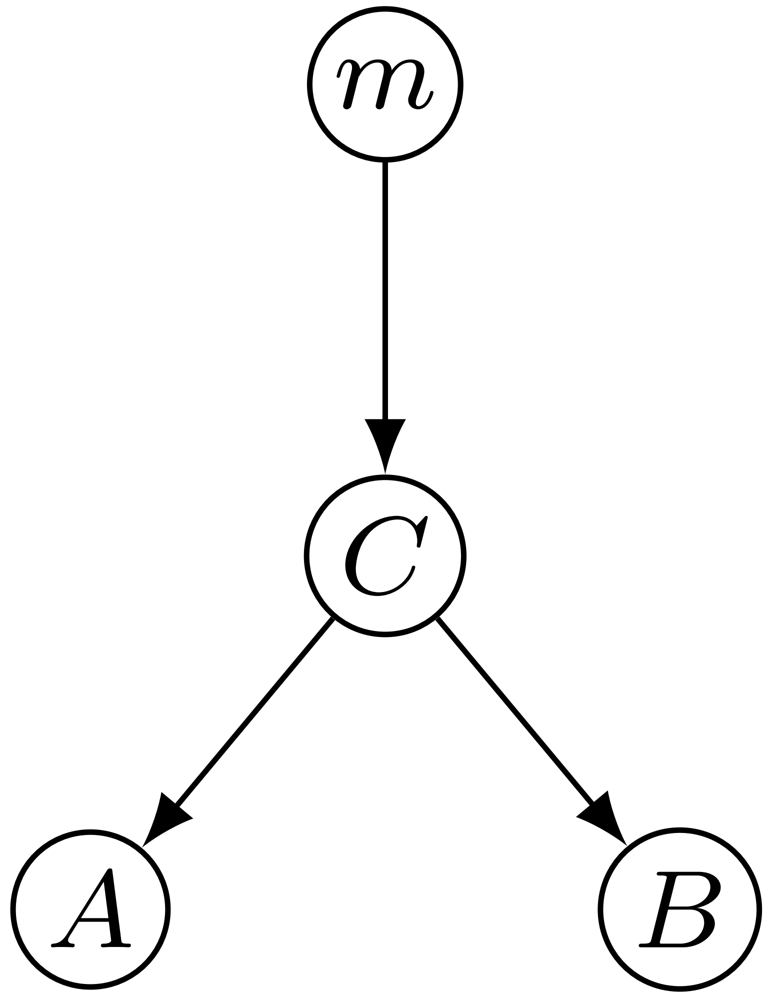
The tuple $(I_A,\ I_B,\ I_{AB})$ has **weak TIP** iff: $I_A \wedge B \implies I_{AB}$

Why tree interpolation property is required?

Version 1

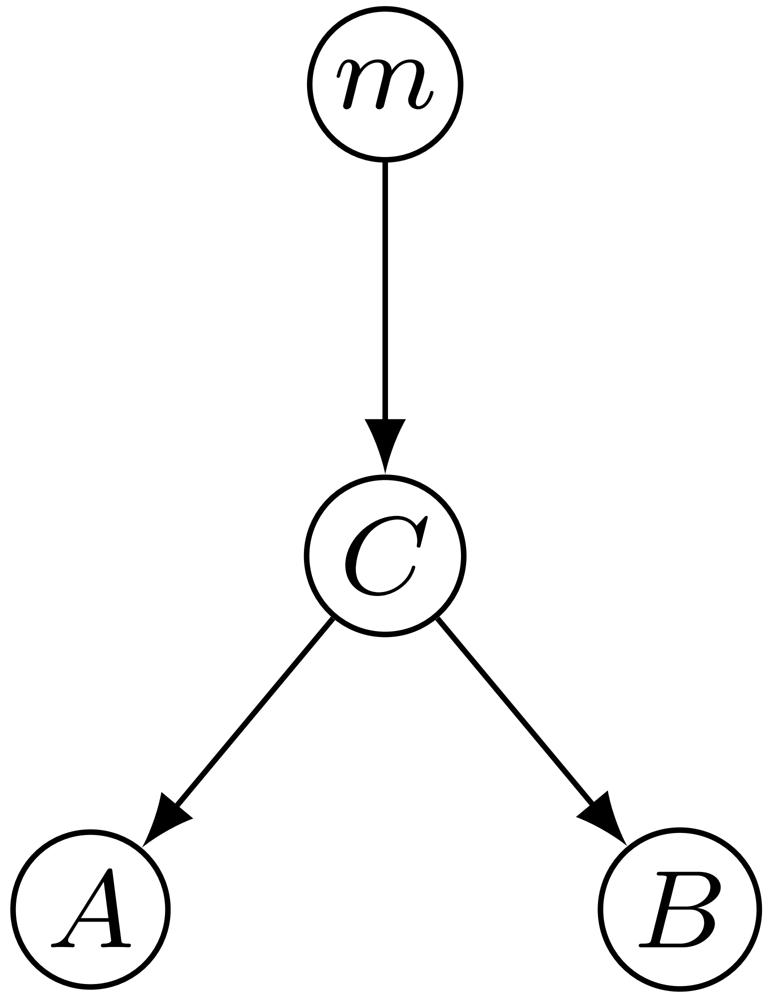
Why tree interpolation property is required?

Version 1



Why tree interpolation property is required?

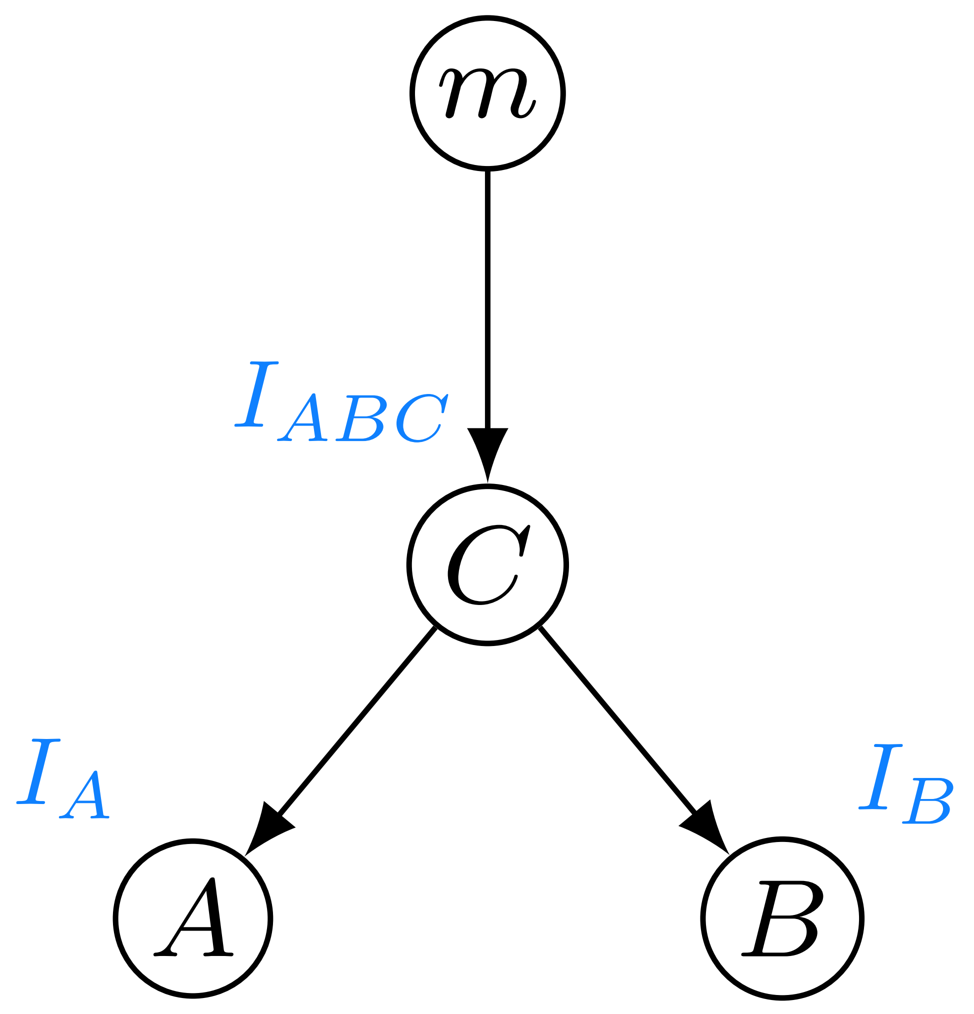
Version 1



✔ Program safe $A \wedge B \wedge C \wedge m \implies \bot$

Why tree interpolation property is required?

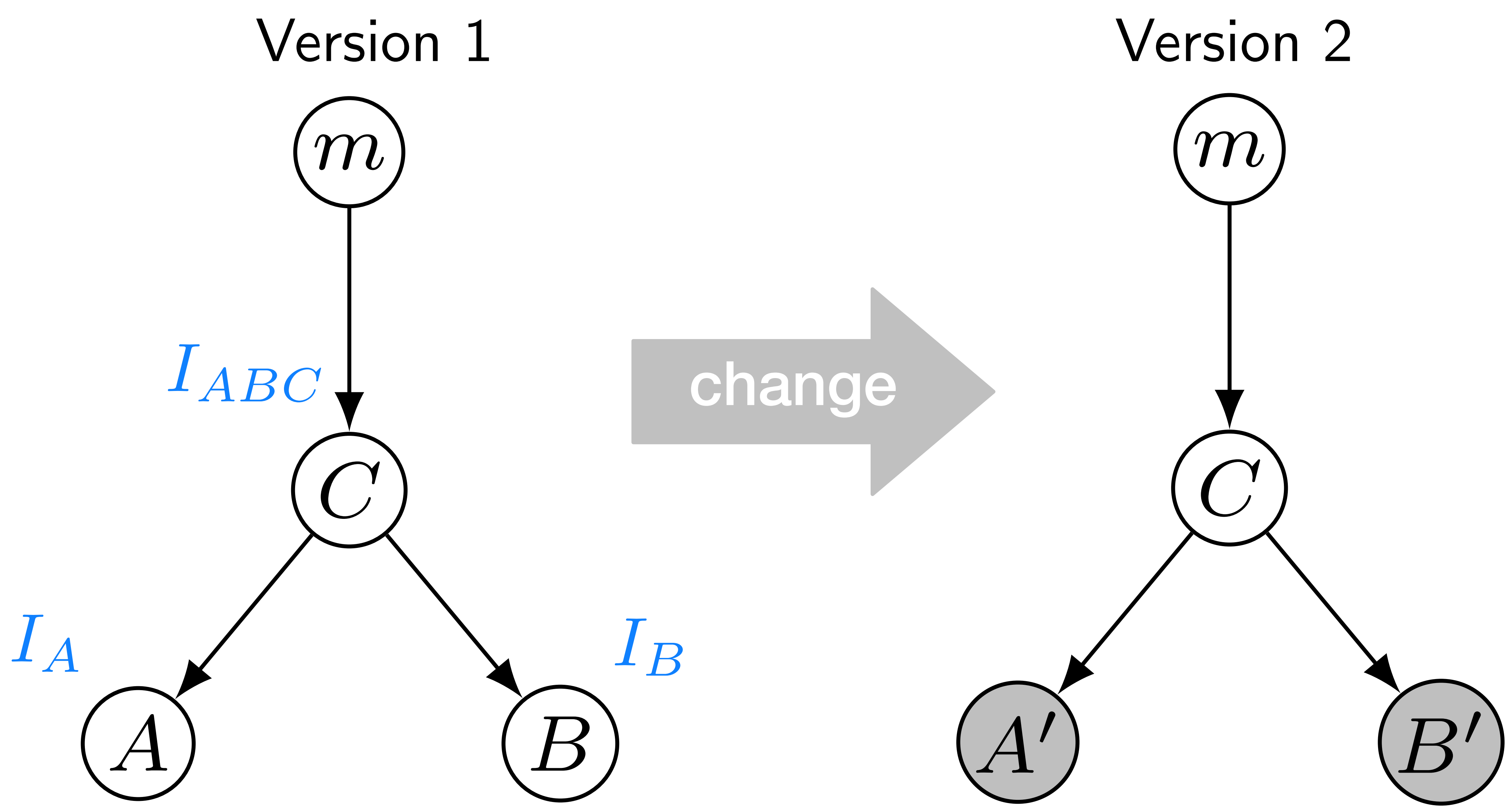
Version 1



✔ Program safe $A \wedge B \wedge C \wedge m \implies \bot$

Extract interpolants $\begin{cases} A \wedge B \wedge C \implies I_{ABC} \\ I_{ABC} \wedge m \implies \bot \end{cases}$

Why tree interpolation property is required?

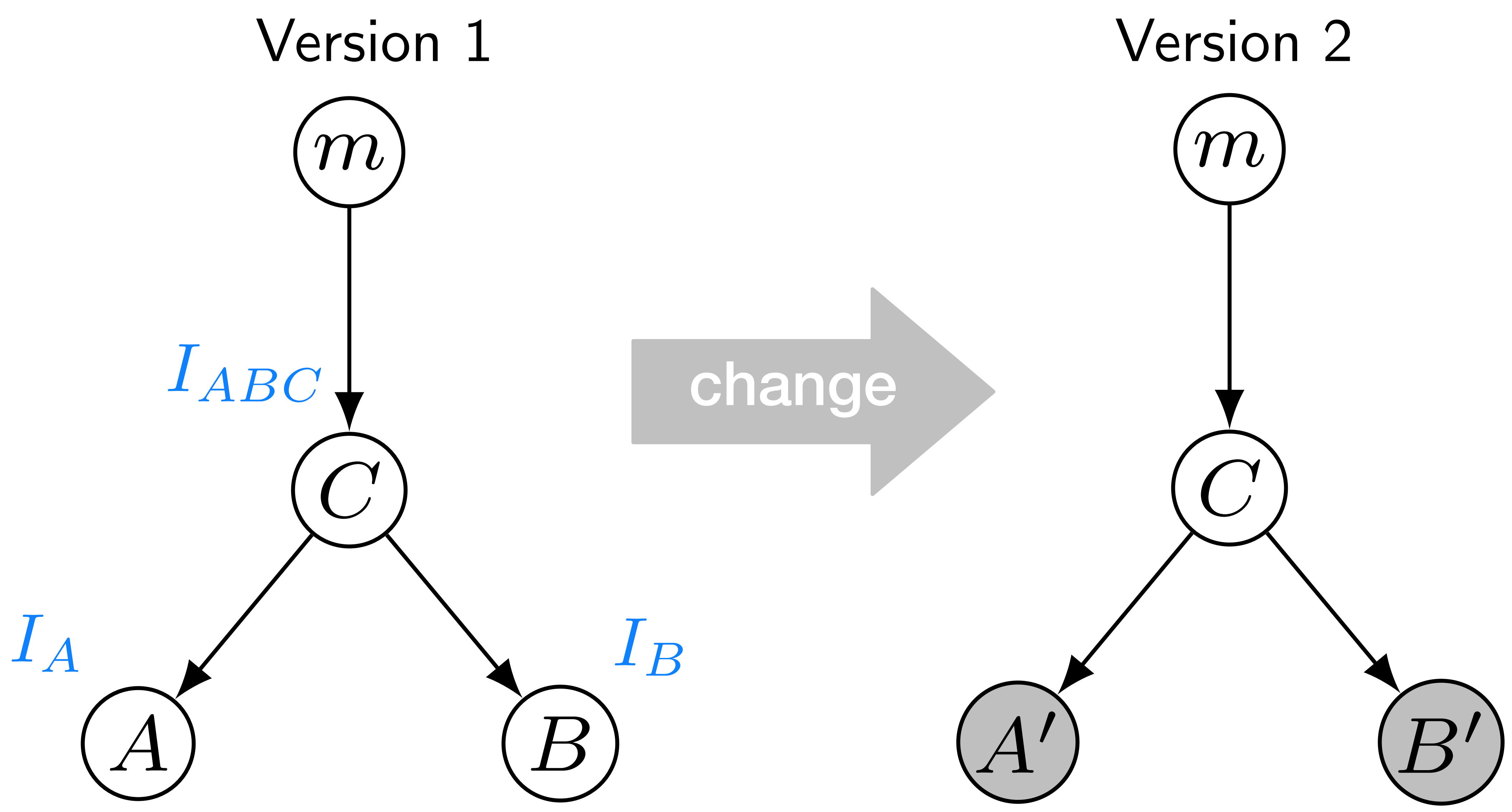


Change arrives $A \rightsquigarrow A'$ $B \rightsquigarrow B'$

✔ Program safe $A \wedge B \wedge C \wedge m \implies \bot$

Extract interpolants $\begin{cases} A \wedge B \wedge C \implies I_{ABC} \\ I_{ABC} \wedge m \implies \bot \end{cases}$

Why tree interpolation property is required?



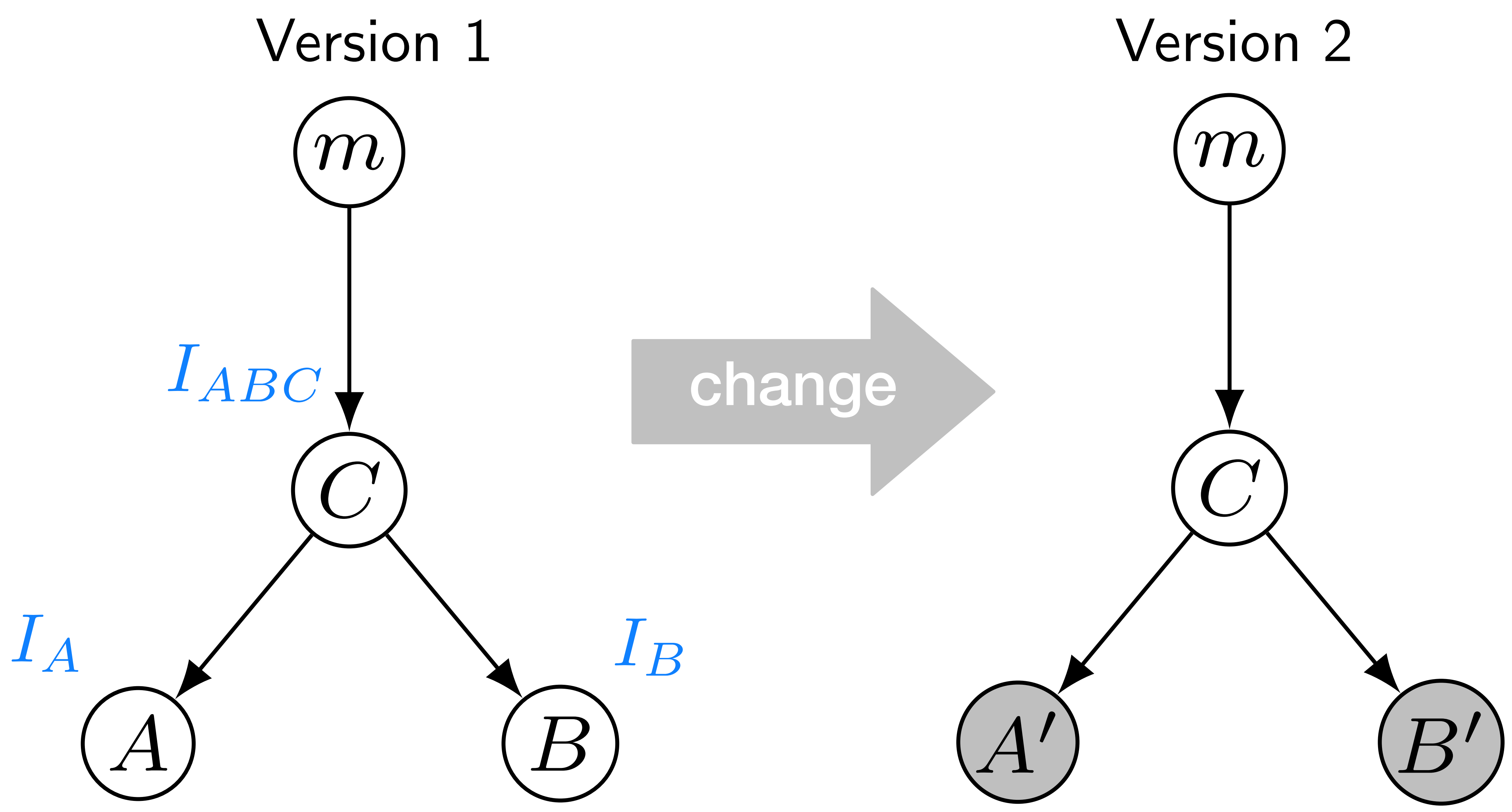
Change arrives $A \rightsquigarrow A'$ $B \rightsquigarrow B'$

Program safe? $A' \wedge B' \wedge C \wedge m \implies ?$

✔ Program safe $A \wedge B \wedge C \wedge m \implies \bot$

Extract interpolants $\begin{cases} A \wedge B \wedge C \implies I_{ABC} \\ I_{ABC} \wedge m \implies \bot \end{cases}$

Why tree interpolation property is required?



Change arrives $A \rightsquigarrow A'$ $B \rightsquigarrow B'$

Program safe? $A' \wedge B' \wedge C \wedge m \implies ?$ ⛔

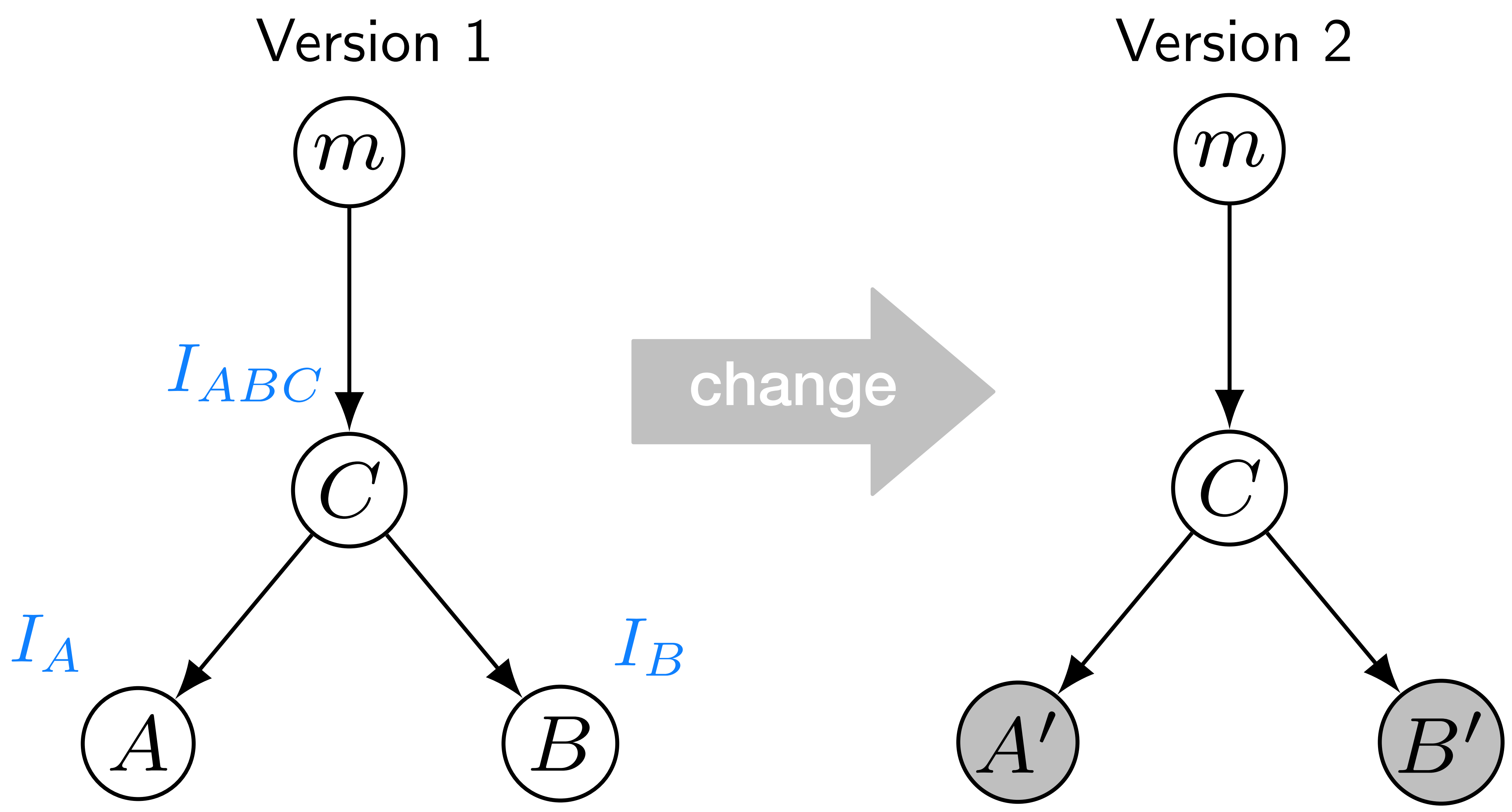
$A' \implies I_A$ ✔

$B' \implies I_B$ ✔

✔ Program safe $A \wedge B \wedge C \wedge m \implies \bot$

Extract interpolants $\begin{cases} A \wedge B \wedge C \implies I_{ABC} \\ I_{ABC} \wedge m \implies \bot \end{cases}$

Why tree interpolation property is required?



Change arrives $A \rightsquigarrow A'$ $B \rightsquigarrow B'$

Program safe? $A' \wedge B' \wedge C \wedge m \implies ?$

$A' \implies I_A$ ✔

$B' \implies I_B$ ✔

$I_A \wedge I_B \wedge C \implies I_{ABC}$ Tree Interpolation Property must hold

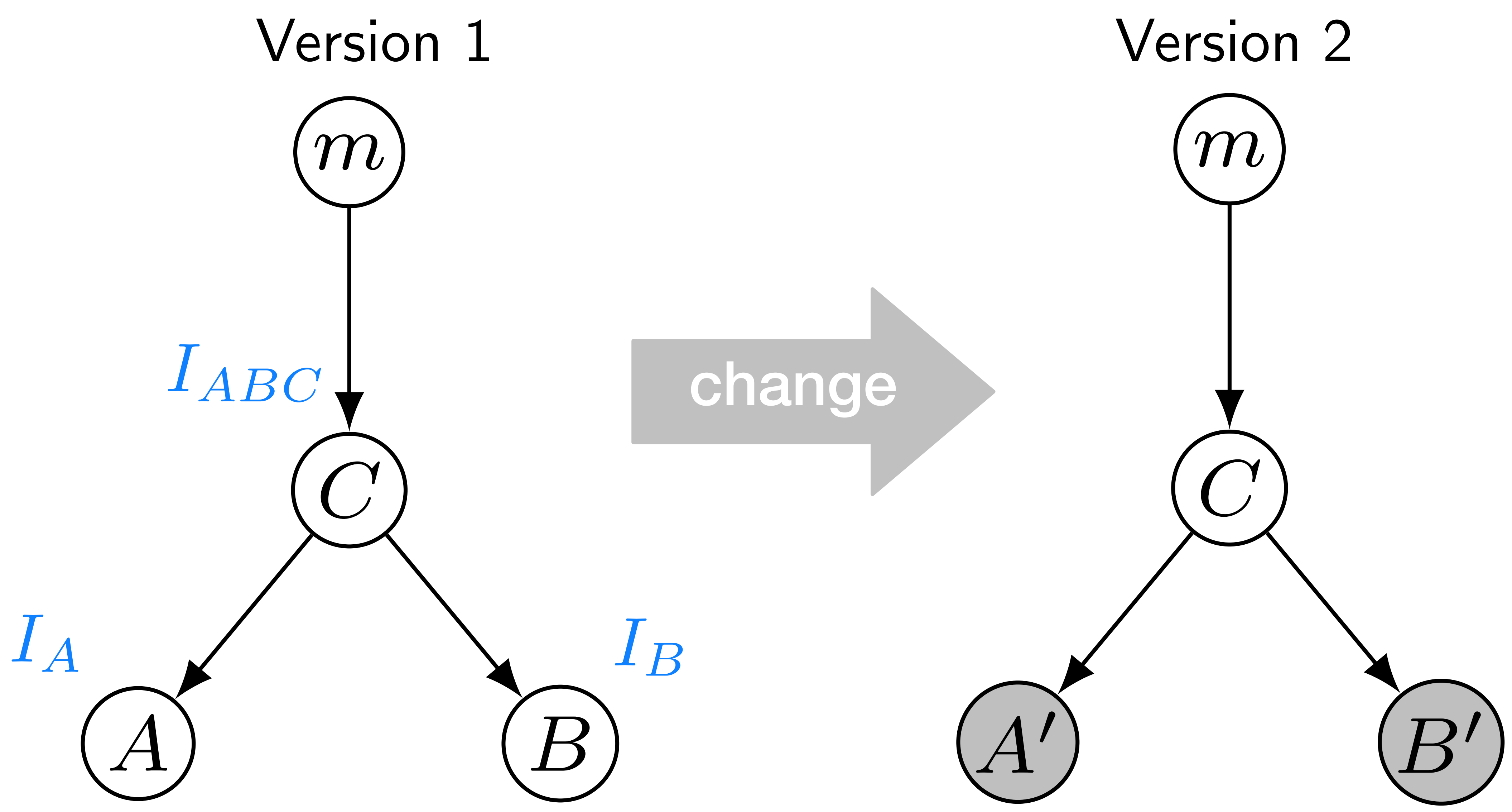
$A' \wedge B' \wedge C \implies I_{ABC}$

$A' \wedge B' \wedge C \wedge m \implies \bot$

✔ Program safe $A \wedge B \wedge C \wedge m \implies \bot$

Extract interpolants $\begin{cases} A \wedge B \wedge C \implies I_{ABC} \\ I_{ABC} \wedge m \implies \bot \end{cases}$

Why tree interpolation property is required?



$\checkmark$ Program safe $\quad A \wedge B \wedge C \wedge m \implies \bot$

Extract interpolants $\begin{cases} A \wedge B \wedge C \implies I_{ABC} \\ I_{ABC} \wedge m \implies \bot \end{cases}$

Change arrives $\quad A \rightsquigarrow A' \quad B \rightsquigarrow B'$

Program safe? $\quad A' \wedge B' \wedge C \wedge m \implies ?$

$A' \implies I_A$ $\checkmark$

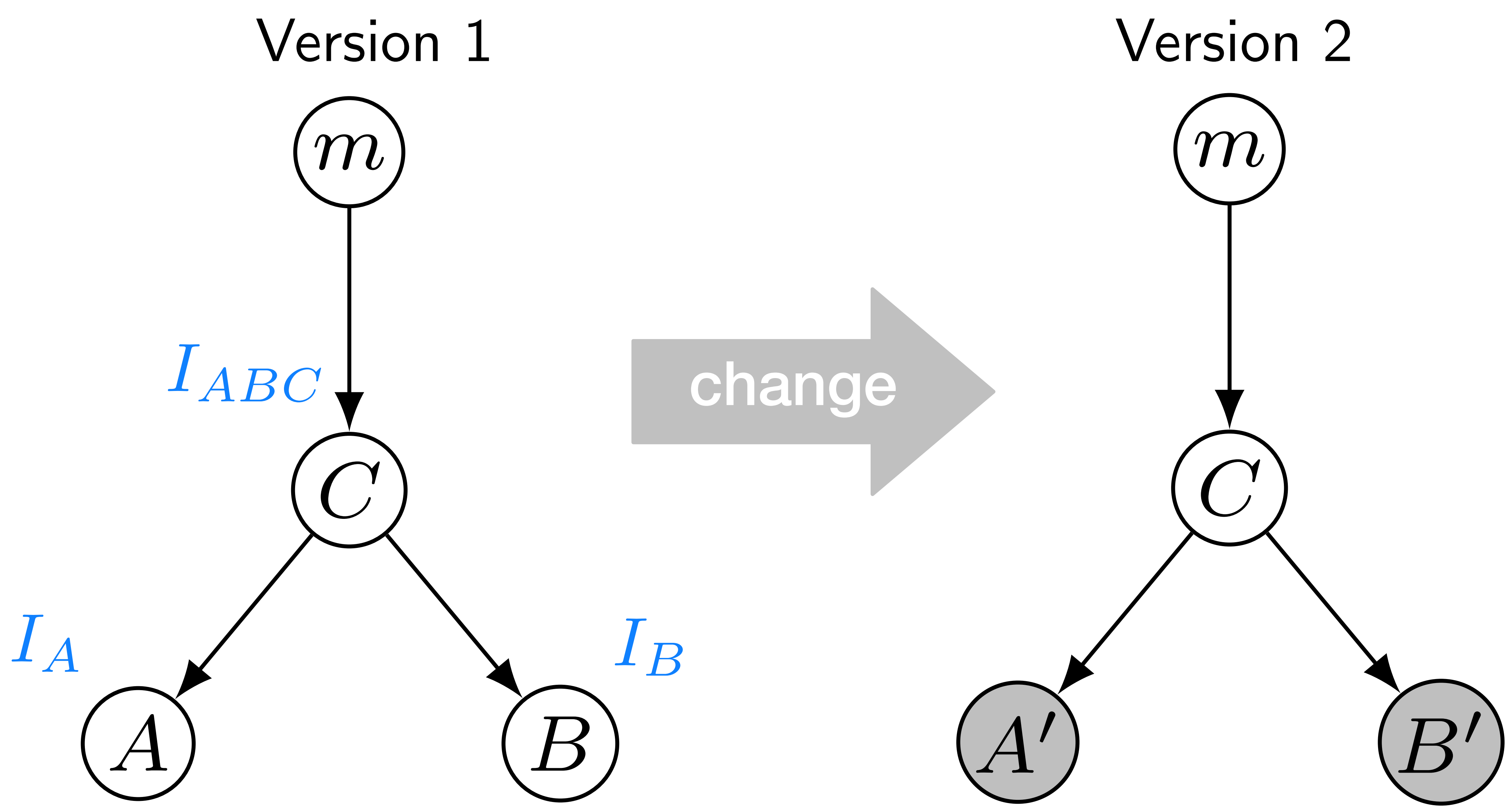
$B' \implies I_B$ $\checkmark$

$I_A \wedge I_B \wedge C \implies I_{ABC}$ Tree Interpolation Property must hold

$A' \wedge B' \wedge C \implies I_{ABC}$

$A' \wedge B' \wedge C \wedge m \implies \bot$

Why tree interpolation property is required?



✔ Program safe $\quad A \wedge B \wedge C \wedge m \implies \bot$

Extract interpolants
$$\begin{cases} A \wedge B \wedge C \implies I_{ABC} \\ I_{ABC} \wedge m \implies \bot \end{cases}$$

Change arrives $\quad A \rightsquigarrow A' \qquad B \rightsquigarrow B'$

Program safe? $\quad A' \wedge B' \wedge C \wedge m \implies ?$

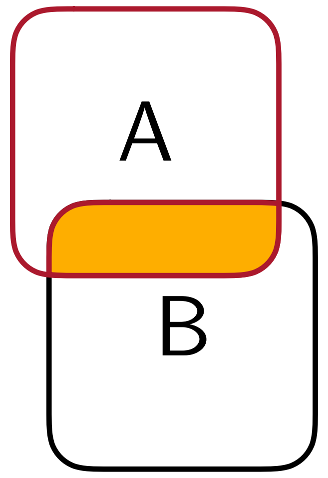
$A' \implies I_A$ ✔

$B' \implies I_B$ ✔

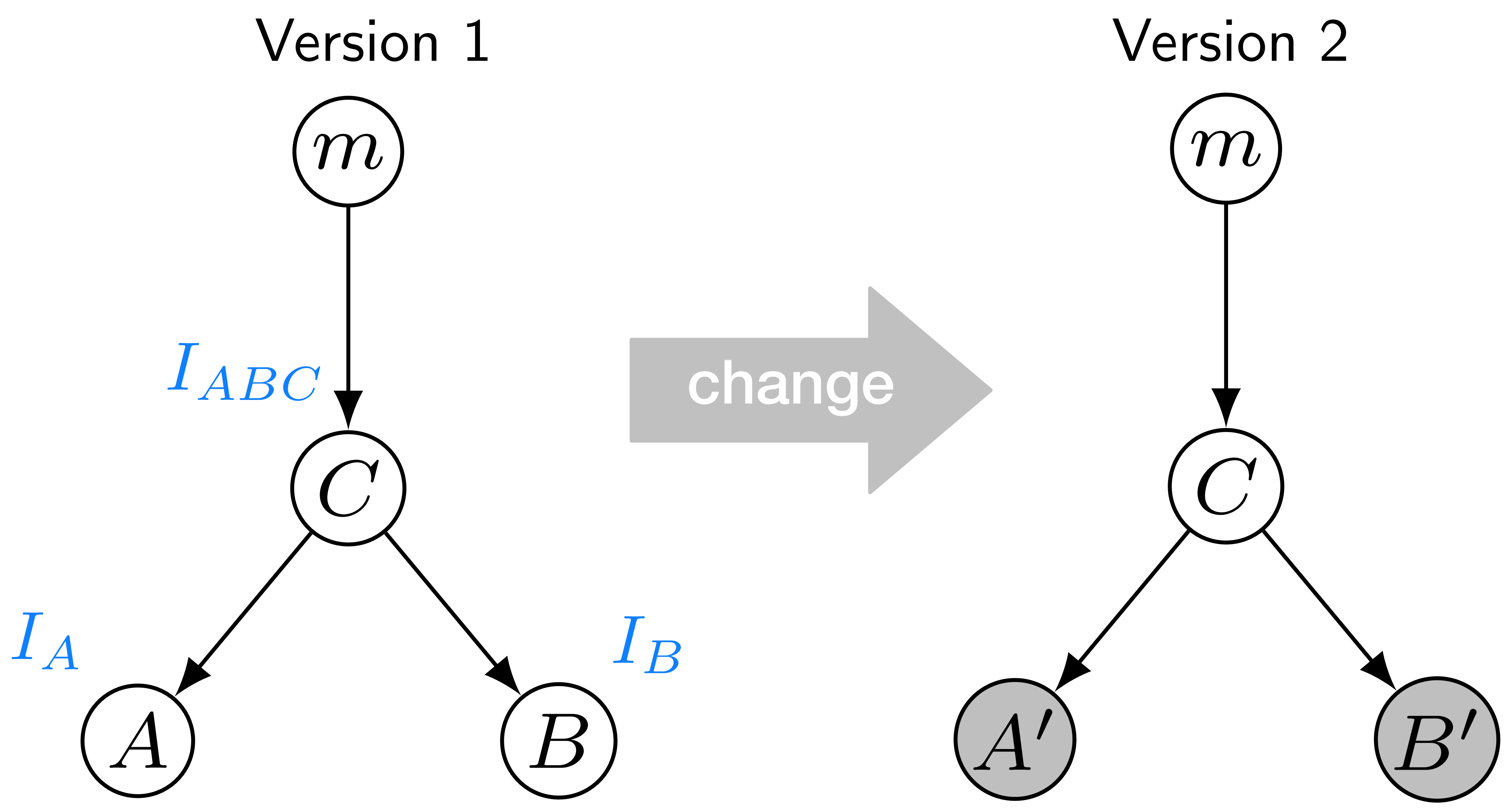
$I_A \wedge I_B \wedge C \implies I_{ABC}$ Tree Interpolation Property must hold

$A' \wedge B' \wedge C \implies I_{ABC}$

$A' \wedge B' \wedge C \wedge m \implies \bot$



Why tree interpolation property is required?



Change arrives $A \rightsquigarrow A'$ $B \rightsquigarrow B'$

Program safe? $A' \wedge B' \wedge C \wedge m \implies ?$

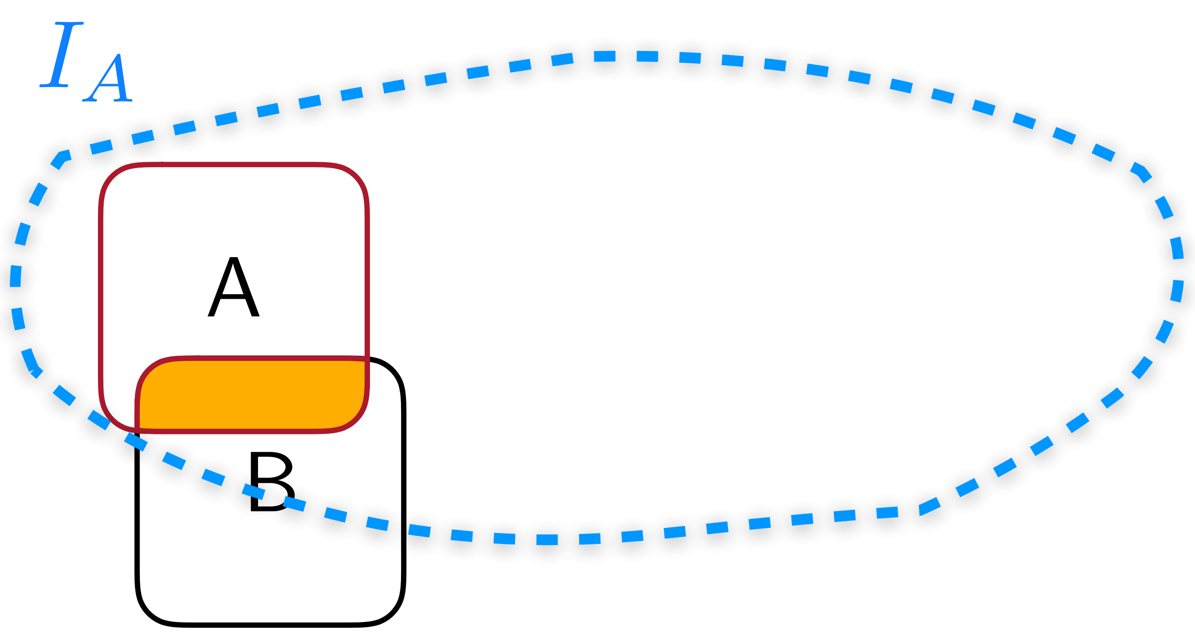
$A' \implies I_A$ ✓

$B' \implies I_B$ ✓

$I_A \wedge I_B \wedge C \implies I_{ABC}$ Tree Interpolation Property must hold

$A' \wedge B' \wedge C \implies I_{ABC}$

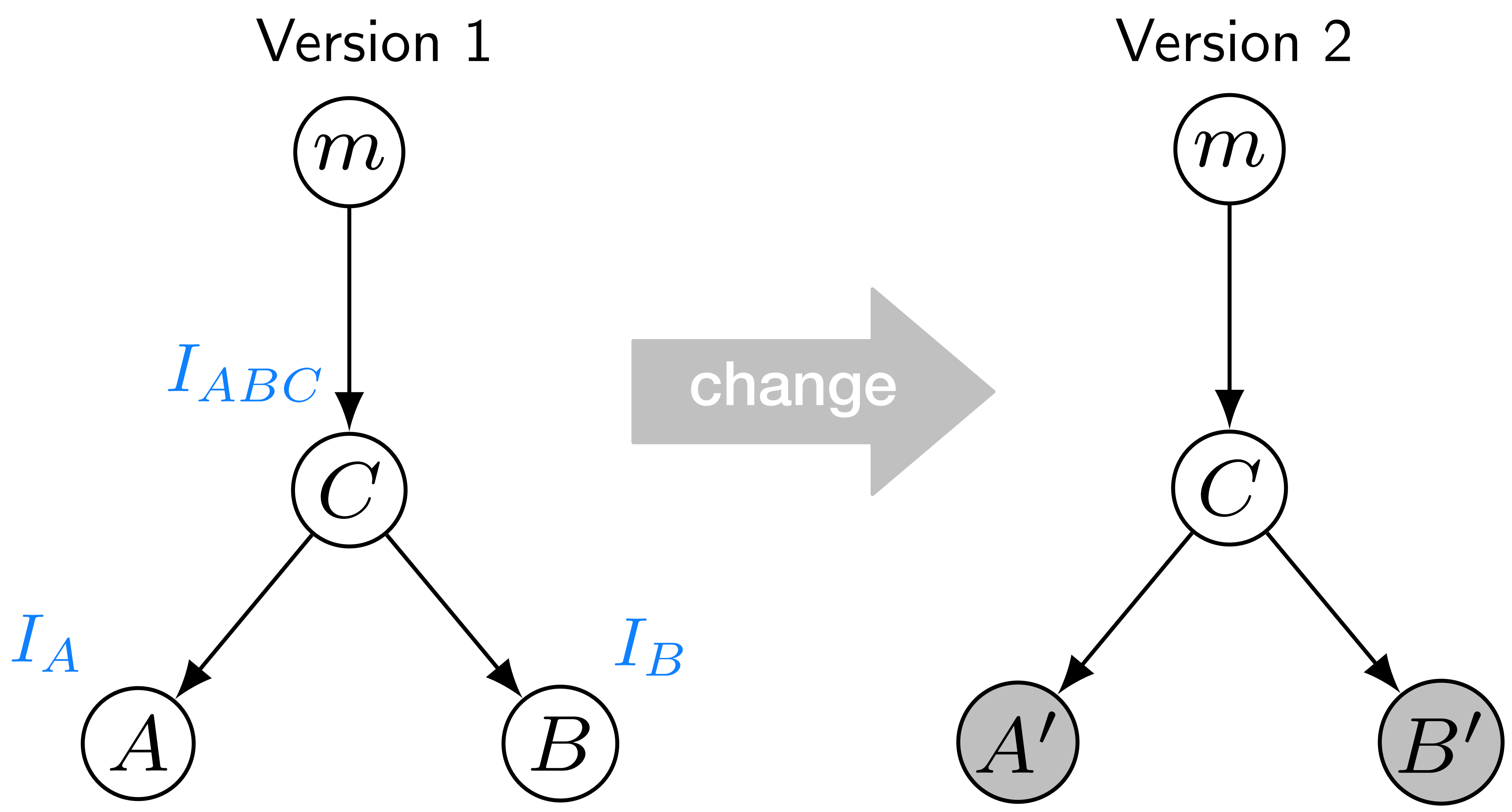
$A' \wedge B' \wedge C \wedge m \implies \bot$



✓ Program safe $A \wedge B \wedge C \wedge m \implies \bot$

Extract interpolants
$$\begin{cases} A \wedge B \wedge C \implies I_{ABC} \\ I_{ABC} \wedge m \implies \bot \end{cases}$$

Why tree interpolation property is required?



✓ Program safe $A \wedge B \wedge C \wedge m \implies \bot$

Extract interpolants $\begin{cases} A \wedge B \wedge C \implies I_{ABC} \\ I_{ABC} \wedge m \implies \bot \end{cases}$

Change arrives $A \rightsquigarrow A'$ $B \rightsquigarrow B'$

Program safe? $A' \wedge B' \wedge C \wedge m \implies ?$

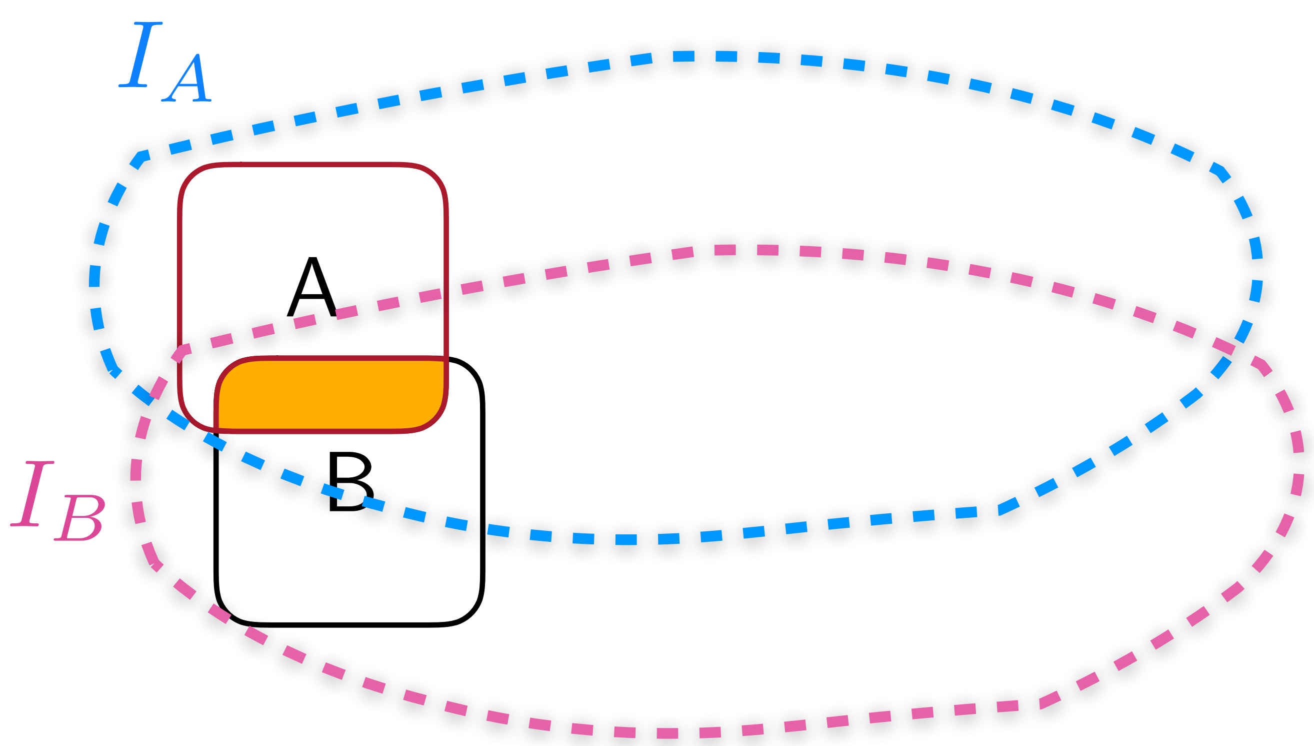
$A' \implies I_A$ ✓

$B' \implies I_B$ ✓

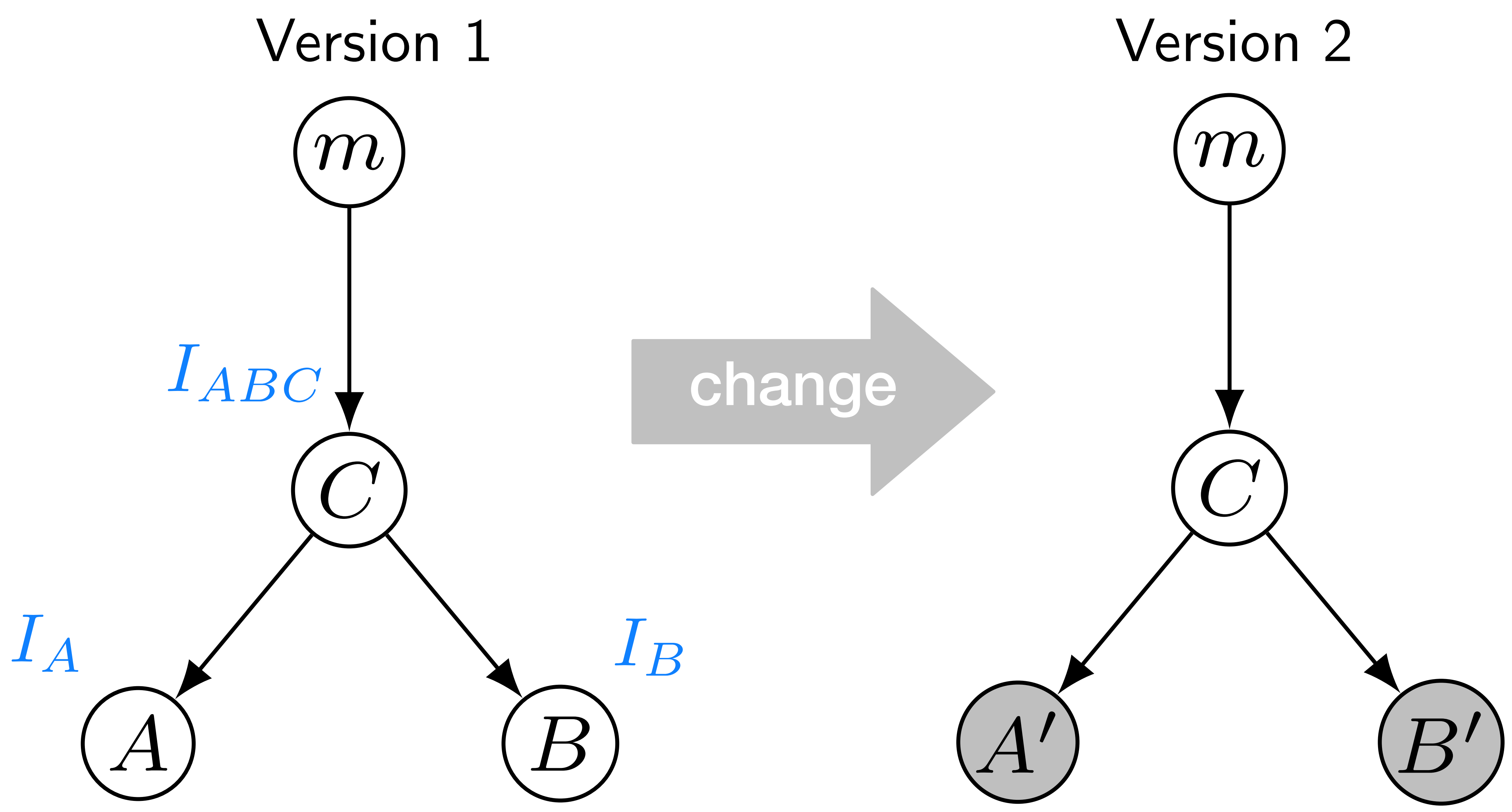
$I_A \wedge I_B \wedge C \implies I_{ABC}$ Tree Interpolation Property must hold

$A' \wedge B' \wedge C \implies I_{ABC}$

$A' \wedge B' \wedge C \wedge m \implies \bot$



Why tree interpolation property is required?



Change arrives $A \rightsquigarrow A'$ $B \rightsquigarrow B'$

Program safe? $A' \wedge B' \wedge C \wedge m \implies ?$

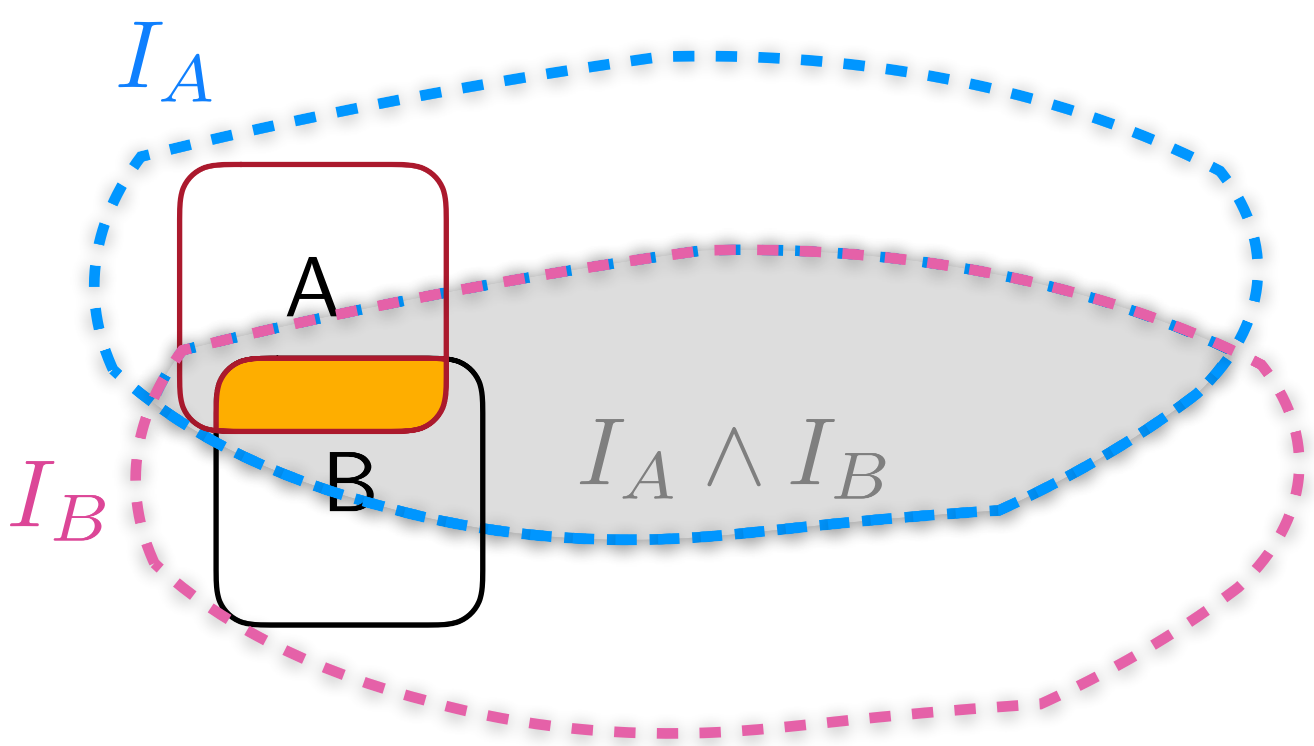
$A' \implies I_A$ ✔

$B' \implies I_B$ ✔

$I_A \wedge I_B \wedge C \implies I_{ABC}$ Tree Interpolation Property must hold

$A' \wedge B' \wedge C \implies I_{ABC}$

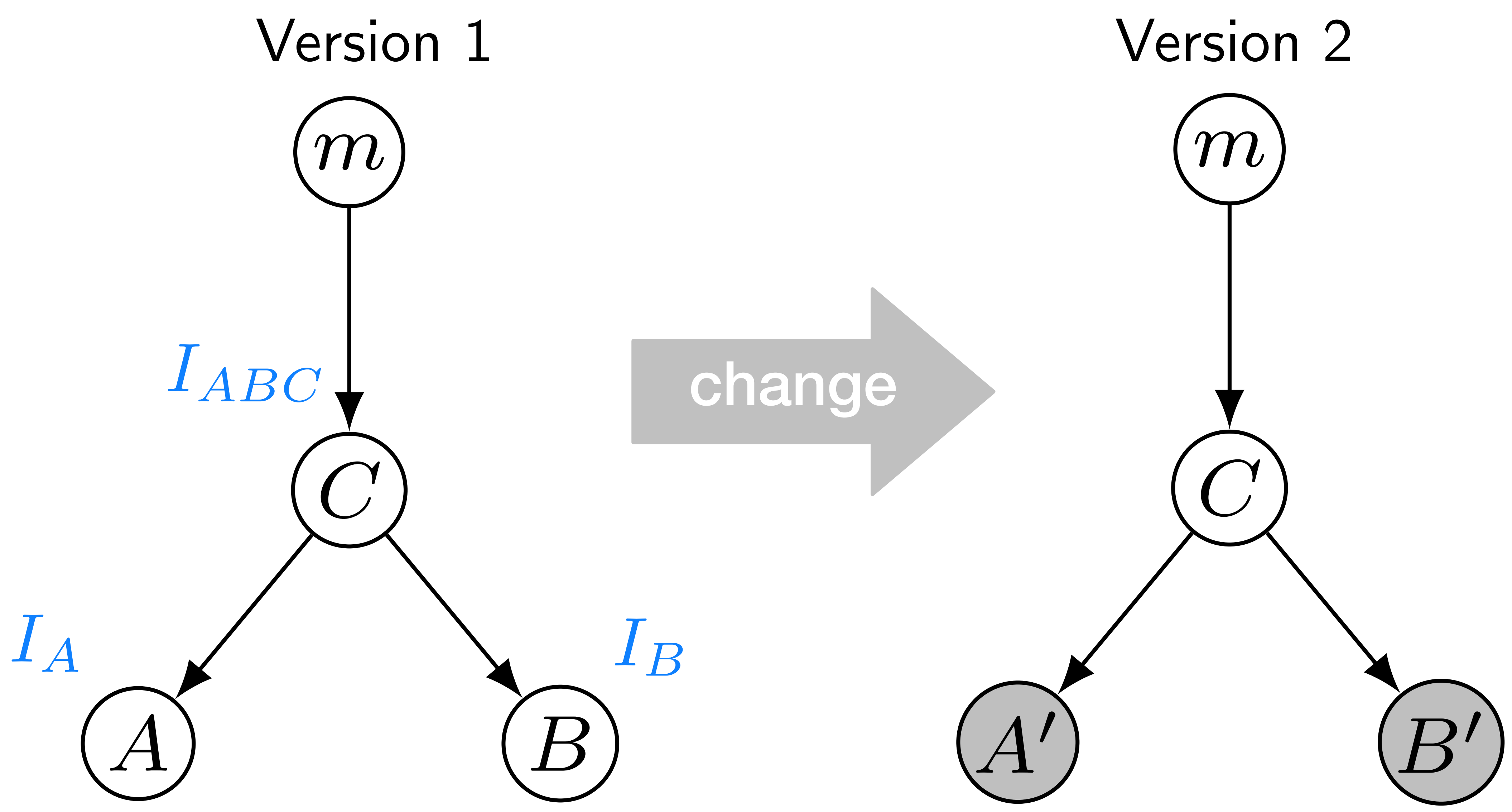
$A' \wedge B' \wedge C \wedge m \implies \bot$



✔ Program safe $A \wedge B \wedge C \wedge m \implies \bot$

Extract interpolants $\begin{cases} A \wedge B \wedge C \implies I_{ABC} \\ I_{ABC} \wedge m \implies \bot \end{cases}$

Why tree interpolation property is required?



Change arrives $A \rightsquigarrow A'$ $B \rightsquigarrow B'$

Program safe? $A' \wedge B' \wedge C \wedge m \implies ?$

$A' \implies I_A$ ✔

$B' \implies I_B$ ✔

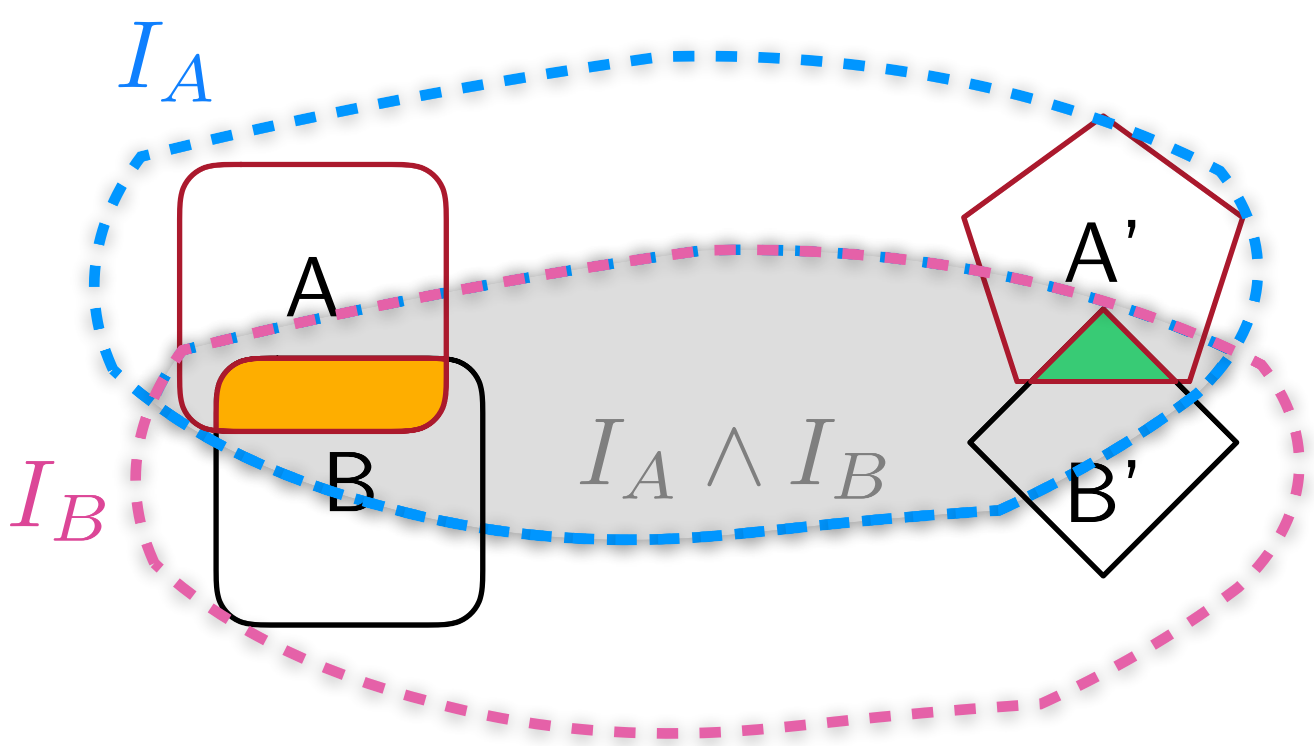
$I_A \wedge I_B \wedge C \implies I_{ABC}$ Tree Interpolation Property must hold

$A' \wedge B' \wedge C \implies I_{ABC}$

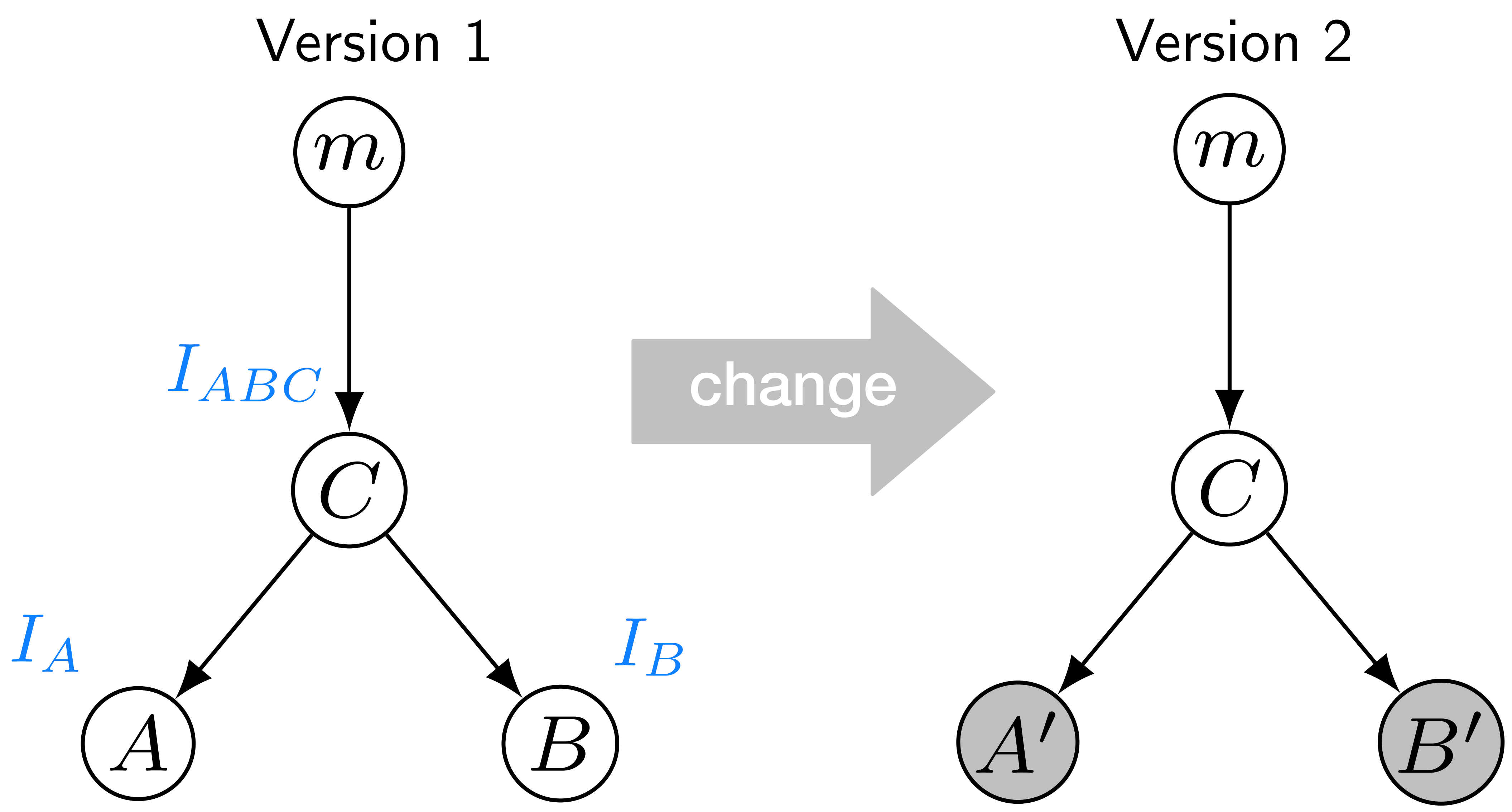
$A' \wedge B' \wedge C \wedge m \implies \bot$

✔ Program safe $A \wedge B \wedge C \wedge m \implies \bot$

Extract interpolants $\begin{cases} A \wedge B \wedge C \implies I_{ABC} \\ I_{ABC} \wedge m \implies \bot \end{cases}$



Why tree interpolation property is required?



Change arrives $A \rightsquigarrow A'$ $B \rightsquigarrow B'$

Program safe? $A' \wedge B' \wedge C \wedge m \implies ?$

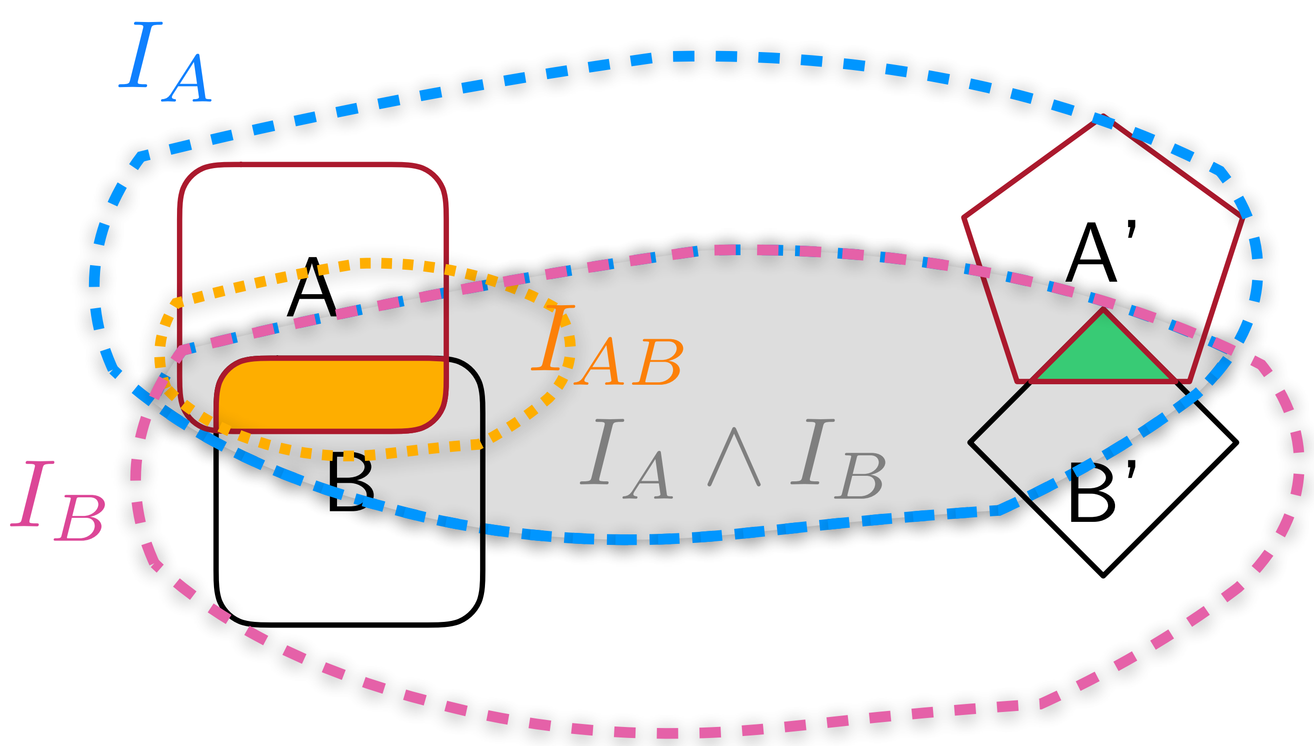
$A' \implies I_A$ ✓

$B' \implies I_B$ ✓

$I_A \wedge I_B \wedge C \implies I_{ABC}$ Tree Interpolation Property must hold

$A' \wedge B' \wedge C \implies I_{ABC}$

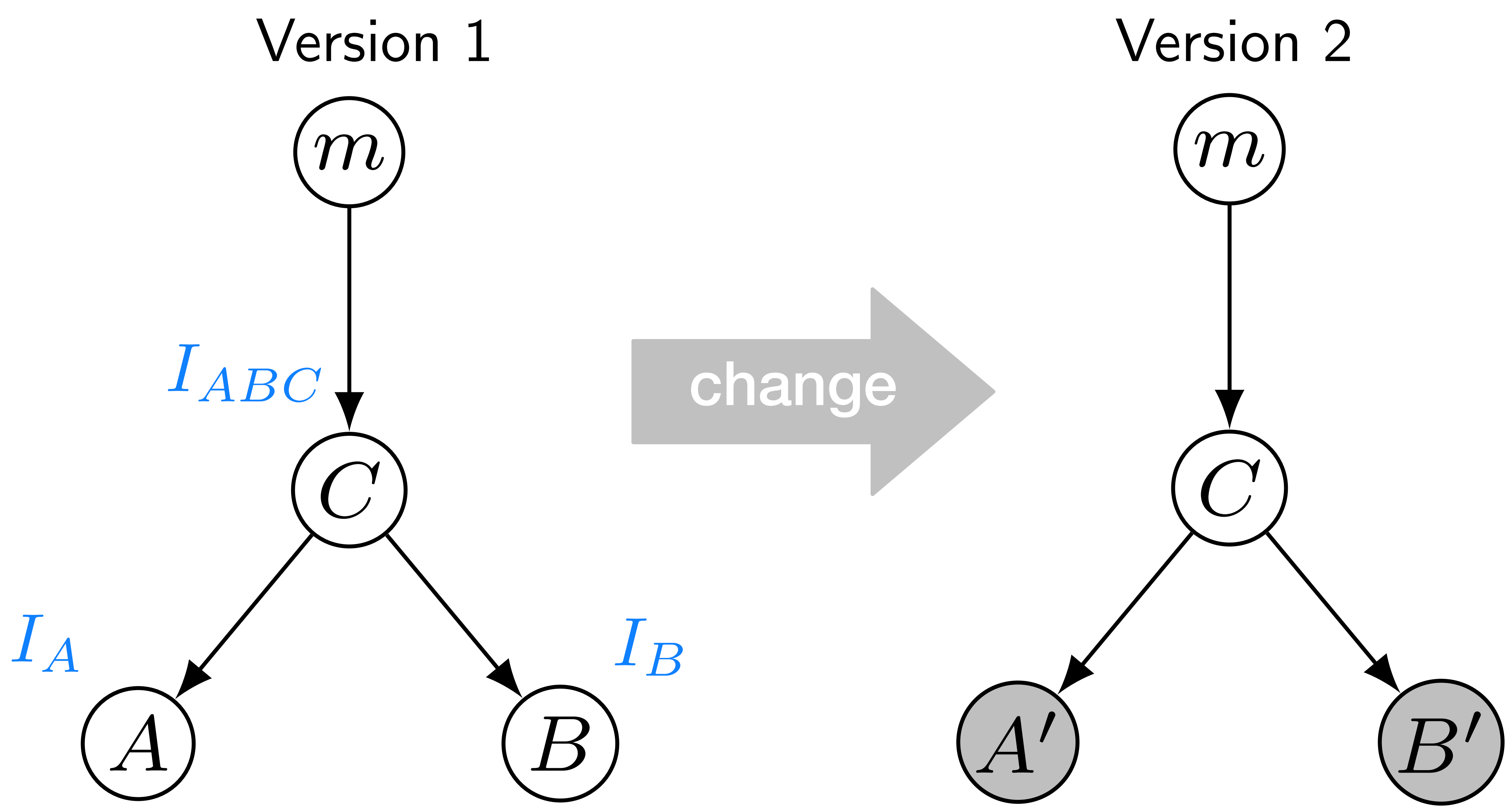
$A' \wedge B' \wedge C \wedge m \implies \bot$



✓ Program safe $A \wedge B \wedge C \wedge m \implies \bot$

Extract interpolants
$$\begin{cases} A \wedge B \wedge C \implies I_{ABC} \\ I_{ABC} \wedge m \implies \bot \end{cases}$$

Why tree interpolation property is required?



Change arrives $A \rightsquigarrow A'$ $B \rightsquigarrow B'$

Program safe? $A' \wedge B' \wedge C \wedge m \implies ?$

$A' \implies I_A$ ✔

$B' \implies I_B$ ✔

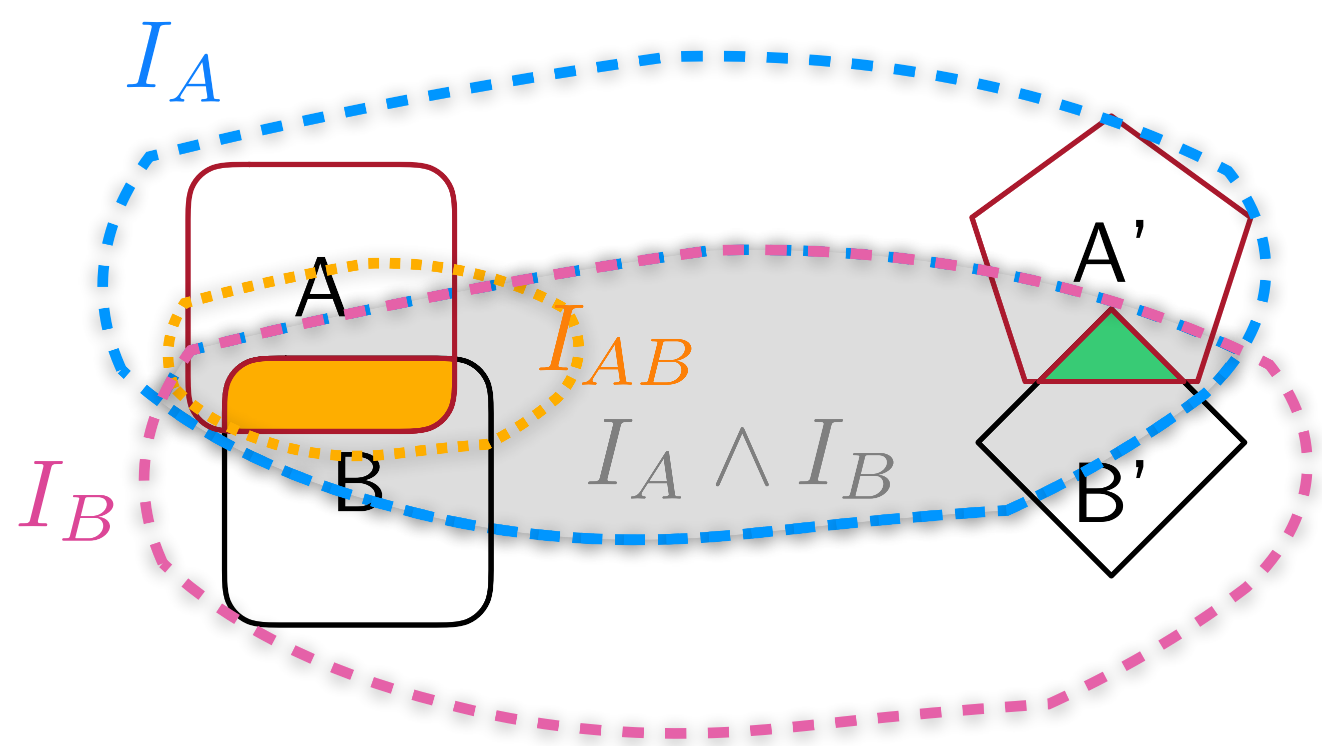
$I_A \wedge I_B \wedge C \implies I_{ABC}$ Tree Interpolation Property must hold

$A' \wedge B' \wedge C \implies I_{ABC}$

$A' \wedge B' \wedge C \wedge m \implies \bot$

✔ Program safe $A \wedge B \wedge C \wedge m \implies \bot$

Extract interpolants $\begin{cases} A \wedge B \wedge C \implies I_{ABC} \\ I_{ABC} \wedge m \implies \bot \end{cases}$



$I_A \wedge I_B \not\Rightarrow I_{AB}$

$A' \wedge B' \not\Rightarrow I_{AB}$ & $A' \wedge B' \wedge C \wedge m \not\Rightarrow ?$

Outline (Contributions)

Outline (Contributions)

- Overview on existing binary interpolation algorithms in LRA

Outline (Contributions)

- **Overview on existing binary interpolation algorithms in LRA**

  1) Farkas

Outline (Contributions)

- **Overview on existing binary interpolation algorithms in LRA**

  1) Farkas

  2) Dual Farkas

Outline (Contributions)

- Overview on existing binary interpolation algorithms in LRA

  1) Farkas

  2) Dual Farkas

  3) Flexible Farkas

Outline (Contributions)

- Overview on existing binary interpolation algorithms in LRA

  1) Farkas

  2) Dual Farkas

  3) Flexible Farkas

  4) Decomposing Farkas

Outline (Contributions)

- **Overview on existing binary interpolation algorithms in LRA**

1) Farkas

2) Dual Farkas

3) Flexible Farkas

4) Decomposing Farkas

5) Dual Decomposing Farkas

Outline (Contributions)

- Overview on existing binary interpolation algorithms in LRA

  1) Farkas

  2) Dual Farkas

  3) Flexible Farkas

  4) Decomposing Farkas

  5) Dual Decomposing Farkas

- Investigate the above algorithms guarantee Tree Interpolation Property (TIP)?

Outline (Contributions)

- **Overview on existing binary interpolation algorithms in LRA**

  1) Farkas

  2) Dual Farkas

  3) Flexible Farkas

  4) Decomposing Farkas

  5) Dual Decomposing Farkas

- **Investigate the above algorithms guarantee Tree Interpolation Property (TIP)?**

  If Yes → Prove it in general!

  If NO → Define a constraint to guarantee TIP!

Outline (Contributions)

- **Overview on existing binary interpolation algorithms in LRA**

  1) Farkas ✓

  2) Dual Farkas ✗

  3) Flexible Farkas ✗

  4) Decomposing Farkas

  5) Dual Decomposing Farkas ✗

- **Investigate the above algorithms guarantee Tree Interpolation Property (TIP)?**

  If Yes → Prove it in general!

  If NO → Define a constraint to guarantee TIP!

Outline (Contributions)

- **Overview on existing binary interpolation algorithms in LRA**

  1) Farkas ✓
  2) Dual Farkas ✗
  3) Flexible Farkas ✗
  4) Decomposing Farkas ✓
  5) Dual Decomposing Farkas ✗

- **Investigate the above algorithms guarantee Tree Interpolation Property (TIP)?**

  If Yes → Prove it in general!

  If NO → Define a constraint to guarantee TIP!

Farkas coefficients

set of **linear inequalities** over real variables

$$\begin{aligned} x_1 &\leq 0 \\ x_2 - x_1 &\leq 0 \\ x_3 - x_1 &\leq 0 \\ -x_2 - x_3 &\leq -1 \end{aligned}$$

Farkas coefficients

$$\begin{aligned} 2\times \quad x_1 &\leq 0 \\ 1\times \quad x_2 - x_1 &\leq 0 \\ 1\times \quad x_3 - x_1 &\leq 0 \\ 1\times \quad -x_2 - x_3 &\leq -1 \\ \hline 0 &\leq -1 \end{aligned}$$

For an unsatisfiable system of linear inequalities Farkas coefficients always exist such that the weighted sum of the system given by the Farkas coefficients —> contradictory inequality

Farkas Interpolant [McMillan 2004]

Ex: Compute interpolant for $(A \wedge B \mid C)$

$$\begin{aligned} x_1 &\leq 0 \\ x_2 - x_1 &\leq 0 \\ x_3 - x_1 &\leq 0 \\ -x_2 - x_3 &\leq -1 \end{aligned}$$

Farkas Interpolant [McMillan 2004]

Ex: Compute interpolant for $(A \wedge B \mid C)$

$$\left.\begin{aligned} x_1 &\leq 0 \\ x_2 - x_1 &\leq 0 \end{aligned}\right\} \text{A}$$

$$\left.\begin{aligned} x_3 - x_1 &\leq 0 \end{aligned}\right\} \text{B}$$

$$\left.\begin{aligned} -x_2 - x_3 &\leq -1 \end{aligned}\right\} \text{C}$$

Farkas Interpolant [McMillan 2004]

Ex: Compute interpolant for $(A \wedge B \mid C)$

$$\left.\begin{aligned} x_1 &\le 0 \\ x_2 - x_1 &\le 0 \end{aligned}\right\} A$$

$$\left.\begin{aligned} x_3 - x_1 &\le 0 \end{aligned}\right\} B$$

$$\left.\begin{aligned} -x_2 - x_3 &\le -1 \end{aligned}\right\} C$$

Farkas coefficients

$$\begin{aligned} 2\times \quad x_1 &\le 0 \\ 1\times \quad x_2 - x_1 &\le 0 \\ 1\times \quad x_3 - x_1 &\le 0 \\ 1\times \quad -x_2 - x_3 &\le -1 \\ \hline 0 &\le -1 \end{aligned}$$

$$\begin{aligned} 2\times \quad x_1 &\le 0 \\ 1\times \quad x_2 - x_1 &\le 0 \\ 1\times \quad x_3 - x_1 &\le 0 \\ 1\times \quad -x_2 - x_3 &\le -1 \\ \hline\hline x_2 + x_3 &\le 0 \end{aligned}$$

Interpolant is the weighted sum of 1st-part of interpolation problem!

Farkas interpolant for $(A \wedge B \mid C)$

Dual Farkas Interpolant

Dual Farkas interpolant for $(A \wedge B \mid C)$ is a negation of the Farkas interpolant for $(C \mid A \wedge B)$:

$$I^F \; := \; x_2 + x_3 \leq 0$$

Farkas Interpolant

$$\overline{I^F} := \neg(C) = x_2 + x_3 < 1$$

Dual Farkas Interpolant

Flexible Farkas Interpolant [Alt et al. HVC'17]

An infinitely many interpolants between **Farkas** and **dual Farkas** interpolants with flexible strength

$$I^F \ := \ x_2 + x_3 \leq 0$$

Farkas Interpolant

$$\overline{I^F} := \neg(C) = x_2 + x_3 < 1$$

Dual Farkas Interpolant

$$I^\alpha \ := \ x_2 + x_3 \leq 1 - \alpha$$

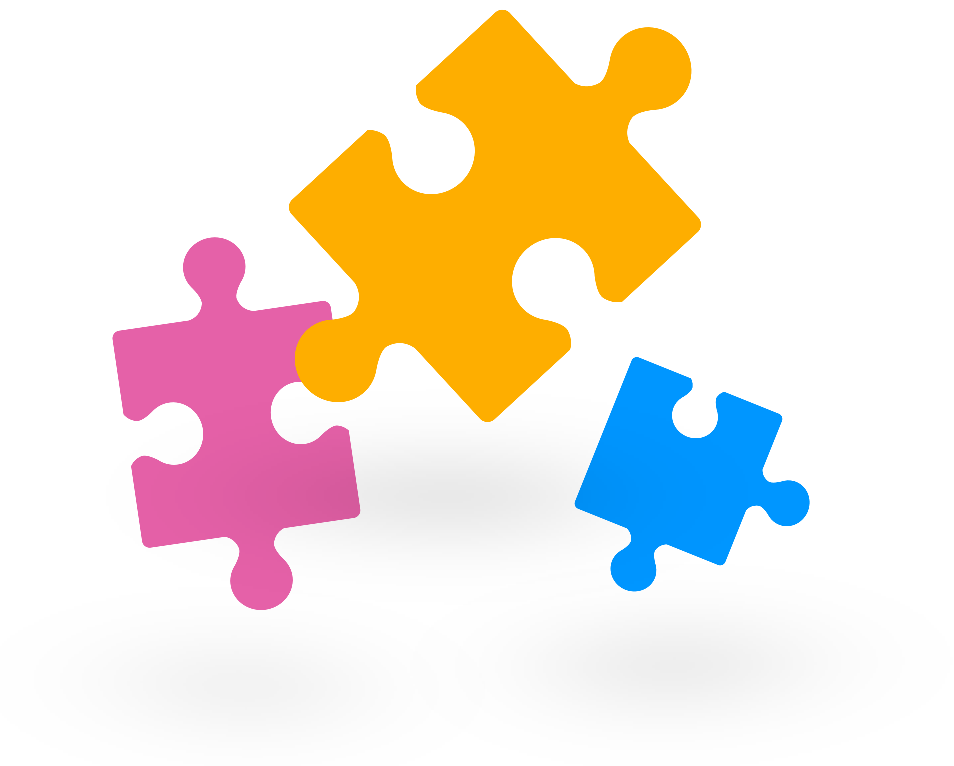
$$0 < \alpha \leq 1$$

Flexible Farkas Interpolant

Decomposed Farkas interpolant [Blicha et al. TACAS'19]

Given a vector of Farkas coefficients, it can be decomposed into a sum of sub-vectors. For e.g.,

$$\begin{pmatrix}2\\1\\1\end{pmatrix} = \begin{pmatrix}1\\1\\0\end{pmatrix} + \begin{pmatrix}1\\0\\1\end{pmatrix}$$



Sub-vectors (decomposition) must be:

1) Non-negative

2) Eliminate the local variables in weighted sums (A-local variables disappear in the interpolant)

Decomposed Farkas interpolant [Blicha et al. TACAS'19]

$$\begin{array}{rr} 2\times & x_1 \le 0 \\ 1\times & x_2 - x_1 \le 0 \\ 1\times & x_3 - x_1 \le 0 \\ \hline & x_2 + x_3 \le 0 \end{array}$$

$$\begin{array}{rr} 1\times & x_1 \le 0 \\ 1\times & x_2 - x_1 \le 0 \\ 0\times & x_3 - x_1 \le 0 \\ \hline & x_2 \le 0 \end{array}$$

$$\begin{array}{rr} 1\times & x_1 \le 0 \\ 0\times & x_2 - x_1 \le 0 \\ 1\times & x_3 - x_1 \le 0 \\ \hline & x_3 \le 0 \end{array}$$

Decomposed Farkas interpolant for $(A \wedge B \mid C)$

$$x_2 \le 0 \;\wedge\; x_3 \le 0$$

Combining LRA & Propositional is tricky ...

- Atoms could belong to several partitions in CNF formula (as opposed to clauses)
- Careless labelling of theory clause can cause issues ...

Combining LRA & Propositional is tricky … (cont'd)

CNF:

$\phi : a \leq b,\ (\overline{a \leq c}) \vee x,\ b \leq c,\ (\overline{a \leq c}) \vee y,\ \bar{x} \vee \bar{y}$

Theory Clause in LRA:

$cl : (\overline{a \leq b}) \vee (\overline{b \leq c}) \vee a \leq c$

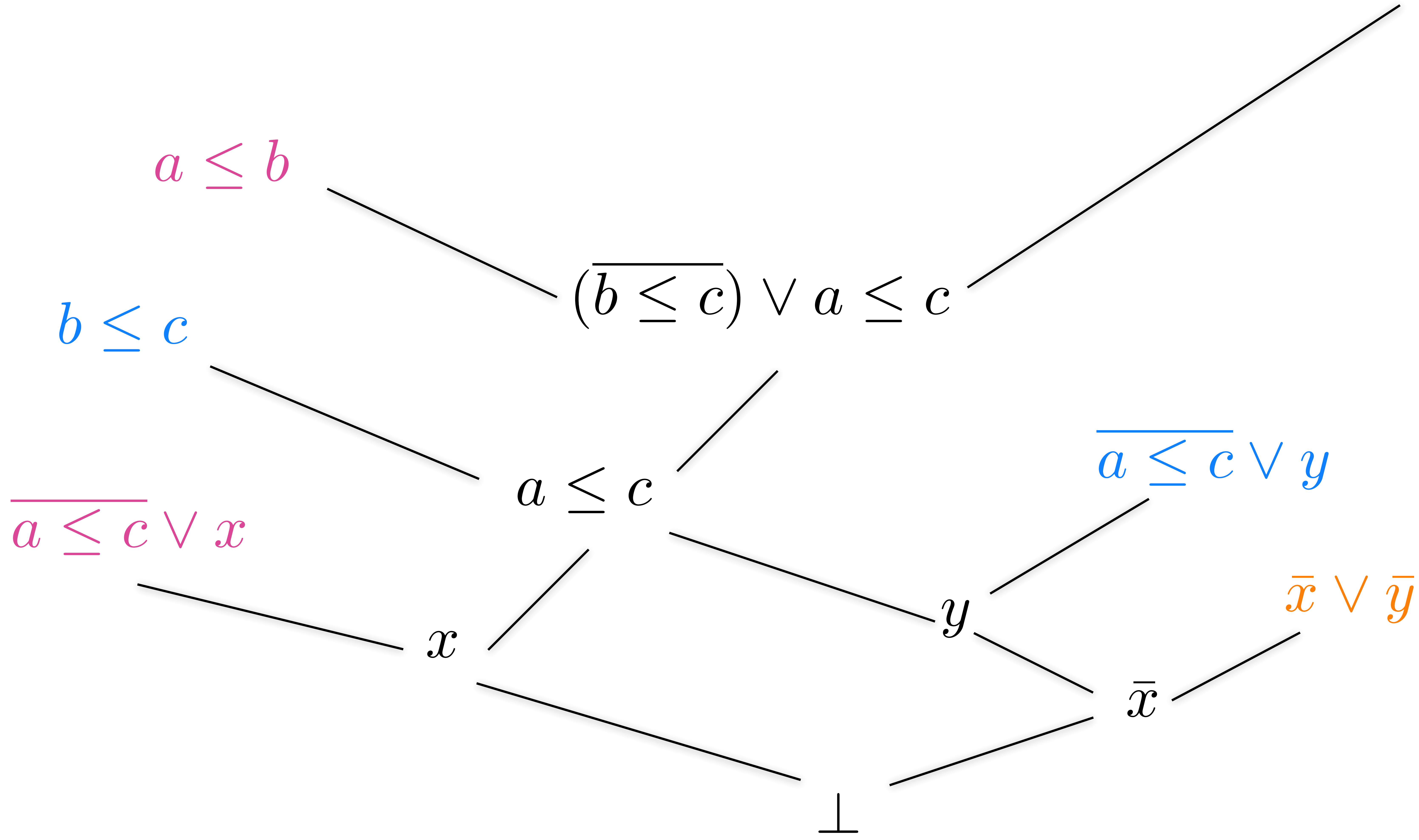
Combining LRA & Propositional is tricky … (cont'd)

CNF:

$\phi : a \leq b,\ (\overline{a \leq c}) \vee x,\ b \leq c,\ (\overline{a \leq c}) \vee y,\ \bar{x} \vee \bar{y}$

Theory Clause in LRA:

$cl : (\overline{a \leq b}) \vee (\overline{b \leq c}) \vee a \leq c$



First, compute **partial interpolants** for the **leaves** of the resolution proof.
Then compute partial interpolants for **inner nodes** as a combination of interpolants of parent nodes.
The **final interpolant** is the partial interpolant of the root of the resolution proof.

Combining LRA & Propositional is tricky … (cont'd)

CNF:

$$\phi : a \le b,\ (\overline{a \le c}) \vee x,\ b \le c,\ (\overline{a \le c}) \vee y,\ \bar{x} \vee \bar{y}$$

Theory Clause in LRA:

$$cl : (\overline{a \le b}) \vee (\overline{b \le c}) \vee a \le c$$

Combining LRA & Propositional is tricky … (cont'd)

CNF:

$$\phi : a \leq b,\ (\overline{a \leq c}) \vee x,\ b \leq c,\ (\overline{a \leq c}) \vee y,\ \bar{x} \vee \bar{y}$$

Partitions: A ($a \leq b,\ (\overline{a \leq c}) \vee x$), B ($b \leq c,\ (\overline{a \leq c}) \vee y$), C ($\bar{x} \vee \bar{y}$)

Theory Clause in LRA:

$$cl : (\overline{a \leq b}) \vee (\overline{b \leq c}) \vee a \leq c$$

Combining LRA & Propositional is tricky … (cont'd)

CNF:

$\phi : a \leq b,\ (\overline{a \leq c}) \vee x,\ b \leq c,\ (\overline{a \leq c}) \vee y,\ \bar{x} \vee \bar{y}$

Partitions: A ($a \leq b,\ (\overline{a \leq c}) \vee x$), B ($b \leq c,\ (\overline{a \leq c}) \vee y$), C ($\bar{x} \vee \bar{y}$)

Theory Clause in LRA:

$cl : (\overline{a \leq b}) \vee (\overline{b \leq c}) \vee a \leq c$

Binary interpolation problem	Binary partitioning of theory clause	Resulting interpolant of cl
$(A \mid B \wedge C)$		
$(B \mid A \wedge C)$		
$(A \wedge B \mid C)$		

Combining LRA & Propositional is tricky ... (cont'd)

CNF:

$$\phi : \underbrace{a \leq b,\ (\overline{a \leq c}) \vee x}_{A},\ \underbrace{b \leq c,\ (\overline{a \leq c}) \vee y}_{B},\ \underbrace{\bar{x} \vee \bar{y}}_{C}$$

Partitions: A B C

Theory Clause in LRA:

$$cl : \underbrace{(\overline{a \leq b})}_{A} \vee \underbrace{(\overline{b \leq c})}_{B} \vee a \leq c$$

Binary interpolation problem	Binary partitioning of theory clause	Resulting interpolant of cl
$(A \mid B \wedge C)$		
$(B \mid A \wedge C)$		
$(A \wedge B \mid C)$		

Combining LRA & Propositional is tricky … (cont'd)

CNF:

$\phi : a \leq b, \ \overline{(a \leq c)} \vee x, \ b \leq c, \ \overline{(a \leq c)} \vee y, \ \bar{x} \vee \bar{y}$

Partitions: A ($a \leq b, \ \overline{(a \leq c)} \vee x$), B ($b \leq c, \ \overline{(a \leq c)} \vee y$), C ($\bar{x} \vee \bar{y}$)

Theory Clause in LRA:

$cl : \overline{(a \leq b)} \vee \overline{(b \leq c)} \vee a \leq c$

A ($\overline{(a \leq b)}$), B ($\overline{(b \leq c)}$), AB? ($a \leq c$)

Binary interpolation problem	Binary partitioning of theory clause	Resulting interpolant of cl
$(A \mid B \wedge C)$		
$(B \mid A \wedge C)$		
$(A \wedge B \mid C)$		

Combining LRA & Propositional is tricky … (cont'd)

CNF:

$\phi : a \leq b, \ \overline{(a \leq c)} \vee x, \ b \leq c, \ \overline{(a \leq c)} \vee y, \ \bar{x} \vee \bar{y}$

Partitions: A B C

Theory Clause in LRA:

$cl : \overline{(a \leq b)} \vee \overline{(b \leq c)} \vee a \leq c$

A B AB?

Binary interpolation problem	Binary partitioning of theory clause	Resulting interpolant of cl
$(A \mid B \wedge C)$	$a \leq c$ in B	$I_A^F = a \leq b$
$(B \mid A \wedge C)$		
$(A \wedge B \mid C)$		

Combining LRA & Propositional is tricky … (cont'd)

CNF:

$\phi : a \leq b, \; \overline{(a \leq c)} \vee x, \; b \leq c, \; \overline{(a \leq c)} \vee y, \; \bar{x} \vee \bar{y}$

Partitions: A ($a \leq b, \overline{(a \leq c)} \vee x$), B ($b \leq c, \overline{(a \leq c)} \vee y$), C ($\bar{x} \vee \bar{y}$)

Theory Clause in LRA:

$cl : \overline{(a \leq b)} \vee \overline{(b \leq c)} \vee a \leq c$

A ($\overline{(a \leq b)}$), B ($\overline{(b \leq c)}$), AB? ($a \leq c$)

Binary interpolation problem	Binary partitioning of theory clause	Resulting interpolant of cl
$(A \mid B \wedge C)$	$a \leq c$ in B	$I_A^F = a \leq b$
$(B \mid A \wedge C)$	$a \leq c$ in A	$I_B^F = b \leq c$
$(A \wedge B \mid C)$	$a \leq c$ in A	$I_{AB}^F = \bot$

Combining LRA & Propositional is tricky … (cont'd)

CNF:

$\phi : a \leq b,\ \overline{(a \leq c)} \vee x,\ b \leq c,\ \overline{(a \leq c)} \vee y,\ \bar{x} \vee \bar{y}$

Partitions: A ($a \leq b,\ \overline{(a \leq c)} \vee x$), B ($b \leq c,\ \overline{(a \leq c)} \vee y$), C ($\bar{x} \vee \bar{y}$)

Theory Clause in LRA:

$cl : \overline{(a \leq b)} \vee \overline{(b \leq c)} \vee a \leq c$

Partitions: A ($\overline{(a \leq b)}$), B ($\overline{(b \leq c)}$), AB? ($a \leq c$)

Binary interpolation problem	Binary partitioning of theory clause	Resulting interpolant of cl
$(A \mid B \wedge C)$	$a \leq c$ in B	$I_A^F = a \leq b$
$(B \mid A \wedge C)$	$a \leq c$ in A	$I_B^F = b \leq c$
$(A \wedge B \mid C)$	$a \leq c$ in A	$I_{AB}^F = \bot$

$$I_A^F \wedge I_B^F \overset{?}{\Longrightarrow} I_{AB}^F$$

$$a \leq b \wedge b \leq c \not\Longrightarrow \bot$$

Violates the strong TIP

Solution: Proper Labeling

to determine the partitioning of theory clauses

$cl : \overline{(a \leq b)} \vee \overline{(b \leq c)} \vee a \leq c$

A B A

Binary interpolation problem	Binary partitioning of theory clause	Resulting interpolant of cl
$(A \mid B \wedge C)$	$a \leq c$ in A	
$(B \mid A \wedge C)$	$a \leq c$ in A	$I_B^F = b \leq c$
$(A \wedge B \mid C)$	$a \leq c$ in A	$I_{AB}^F = \bot$

Solution: Proper Labeling

to determine the partitioning of theory clauses

A partitioning of every theory clause should be **fixed** beforehand and stay fixed for all interpolation queries for the **same unsatisfiable formula**.

$cl : \overline{(a \leq b)} \vee \overline{(b \leq c)} \vee a \leq c$

A B A

Binary interpolation problem	Binary partitioning of theory clause	Resulting interpolant of cl
$(A \mid B \wedge C)$	$a \leq c$ in A	
$(B \mid A \wedge C)$	$a \leq c$ in A	$I_B^F = b \leq c$
$(A \wedge B \mid C)$	$a \leq c$ in A	$I_{AB}^F = \bot$

Solution: Proper Labeling

to determine the partitioning of theory clauses

$$cl : \overline{(a \leq b)} \vee \overline{(b \leq c)} \vee a \leq c$$

A B A

A partitioning of every theory clause should be **fixed** beforehand and stay fixed for all interpolation queries for the **same unsatisfiable formula**.

Binary interpolation problem	Binary partitioning of theory clause	Resulting interpolant of cl
$(A \mid B \wedge C)$	$a \leq c$ in A	
$(B \mid A \wedge C)$	$a \leq c$ in A	$I_B^F = b \leq c$
$(A \wedge B \mid C)$	$a \leq c$ in A	$I_{AB}^F = \bot$

Solution: Proper Labeling

to determine the partitioning of theory clauses

A partitioning of every theory clause should be **fixed** beforehand and stay fixed for all interpolation queries for the **same unsatisfiable formula**.

$cl : \overline{(a \leq b)} \vee \overline{(b \leq c)} \vee a \leq c$

A B A

Binary interpolation problem	Binary partitioning of theory clause	Resulting interpolant of cl
$(A \mid B \wedge C)$	$a \leq c$ in A	$I_A^F = a > c$
$(B \mid A \wedge C)$	$a \leq c$ in A	$I_B^F = b \leq c$
$(A \wedge B \mid C)$	$a \leq c$ in A	$I_{AB}^F = \bot$

Solution: Proper Labeling

to determine the partitioning of theory clauses

A partitioning of every theory clause should be **fixed** beforehand and stay fixed for all interpolation queries for the **same unsatisfiable formula**.

$$cl : \underbrace{\overline{(a \le b)}}_{A} \vee \underbrace{\overline{(b \le c)}}_{B} \vee \underbrace{a \le c}_{A}$$

Binary interpolation problem	Binary partitioning of theory clause	Resulting interpolant of cl
$(A \mid B \wedge C)$	$a \le c$ in A	$I_A^F = a > c$
$(B \mid A \wedge C)$	$a \le c$ in A	$I_B^F = b \le c$
$(A \wedge B \mid C)$	$a \le c$ in A	$I_{AB}^F = \bot$

$$I_A^F \wedge I_B^F \implies I_{AB}^F$$

$$a > c \wedge b \le c \implies \bot$$

Strong TIP holds for FARKAS interpolants!

TIP in Farkas Algorithm (Proof by Induction)

Theorem 1 (strong TI in Farkas interpolation). *Let $X \wedge Y \wedge Z$ be an unsatisfiable partitioned CNF formula in* LRA *and let $\mathbb{P}$ be its properly labeled resolution refutation. Let Itp^{P+F} denote the interpolation algorithm that uses Pudlák's algorithm for the propositional part and Farkas algorithm for the theory clauses. Let I_X, I_Y, and I_{XY} be the binary interpolants $Itp^{P+F}(X \,|\, Y \wedge Z)$, $Itp^{P+F}(Y \,|\, X \wedge Z)$, and $Itp^{P+F}(X \wedge Y \,|\, Z)$, respectively. Then $(I_X \wedge I_Y) \implies$*

Theorem 2 (generalizing strong TI in Farkas interpolation). *Let $X_1 \wedge \ldots \wedge X_n \wedge Z$, $n \geq 2$, be an unsatisfiable partitioned CNF formula in* LRA *and let $\mathbb{P}$ be its properly labeled resolution refutation. Let Itp^{P+F} denote the interpolation algorithm that uses Pudlák's algorithm for the propositional part and Farkas algorithm for the theory clauses. Let I_{X_1}, …, I_{X_n}, and $I_{X_1 \ldots X_n}$ be the binary interpolants $Itp^{P+F}(X_1 \,|\, X_2 \wedge \ldots \wedge X_n \wedge Z)$, …, $Itp^{P+F}(X_n \,|\, X_1 \wedge \ldots \wedge X_{n-1} \wedge Z)$ and $Itp^{P+F}(X_1 \wedge \ldots \wedge X_n \,|\, Z)$, respectively. Then $(I_{X_1} \wedge \ldots \wedge I_{X_n}) \implies I_{X_1 \ldots X_n}$, i.e., the tuple $(I_{X_1}, \ldots, I_{X_n}, I_{X_1 \ldots X_n})$ has the strong tree-interpolation property.*

Corollary 1. *$TItp_{Itp^{P+F}}$ is a tree interpolation algorithm, that is, it computes tree interpolants.*

TIP in Farkas Algorithm (Proof by Induction)

Theorem 1 (strong TI in Farkas interpolation). *Let $X \wedge Y \wedge Z$ be an unsatisfiable partitioned CNF formula in* LRA *and let* $\mathbb{P}$ *be its pro[...] labeled resolution refutation. Let* Itp^{P+F} *denote the interpol[...] uses Pudlák's algorithm for the propositional par[...] ory clauses. Let* I_X, I_Y, *and* $I_{X[...]}$ [...] $)$, $Itp^{P+F}(Y \,|\, X \wedge Z)$ [...] $\Rightarrow$

Theorem 2 (ge[...] $\ldots \wedge X_n \wedge Z$, $n \geq 2$ [...] *be its properly lab[...] algorithm that us[...] algorithm for the theory clauses.* [...] *interpolants* $Itp^{P+F}(X_1 \,|\, X_2 \wedge \ldots \wedge X$[...]$Z)$, $\ldots$, [...] *and* $Itp^{P+F}(X_1 \wedge \ldots \wedge X_n \,|\, Z)$, *respe[...]ly. Then* $(I_X$[...]$\ldots \wedge I_{X_n}$[...] *i.e., the tuple* $(I_{X_1}, \ldots, I_{X_n}, I_{X_1 \ldots X_n})$ [...]*ee-interpolation prop*[...]

Corollary 1. $TItp_{Itp^{P+F}}$ *is a tree interpolation algorithm, that is, it computes tree interpolants.*

Theorem : strong/weak TIP in Farkas Interpolation Algorithm is always guaranteed for an arbitrary number of partitions

Negative results of guaranteeing TIP

The following binary interpolation algorithms cannot be used as a basis of a tree interpolation algorithm:

- ☹ **Dual Farkas** interpolation algorithm
- ☹ **Flexible Farkas** interpolation algorithm
- ☹ **Dual decomposing Farkas** interpolation algorithm

Outline (Open Issues and Contributions)

- Overview on different binary interpolation algorithms in LRA

1) Farkas ✓
2) Dual Farkas ✗
3) Flexible Farkas ✗
4) Decomposing Farkas ✓
5) Dual Decomposing Farkas ✗

Outline (Open Issues and Contributions)

- Overview on different **binary interpolation algorithms** in LRA

1) Farkas ✓
2) Dual Farkas ✗
3) Flexible Farkas ✗
4) Decomposing Farkas ✓
5) Dual Decomposing Farkas ✗

TIP in decomposed interpolants

TIP is not guaranteed in general for the decomposed interpolants ... ☹️

... BUT

TIP in decomposed interpolants

TIP is not guaranteed in general for the decomposed interpolants ... ☹

... BUT

We can define a **constraint** on the decompositions that guarantees the TIP! 🙂

Constraints on decompositions to guarantee TIP

Monotonicity Condition: The inequalities resulting from **supersystem's** decomposition must be logically covered by the inequalities of **subsystem's** decomposition

How to achieve monotonic decompositions? By **Gradual Decomposition**!

Gradual Decomposition

A method to restrict possible decompositions used by the decomposing Farkas algorithm

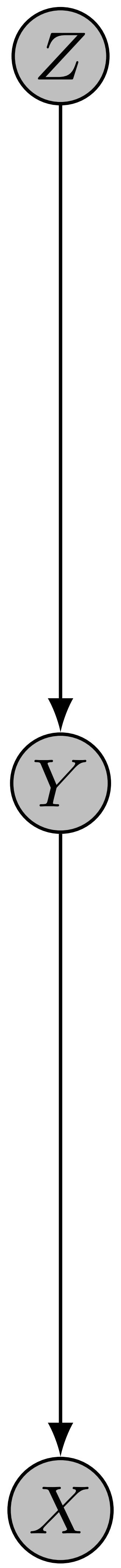
This ensures the requirements of tree interpolant

$(X \wedge Y \wedge Z \mid \top)$
$(X \wedge Y \mid Z)$
$(X \mid Y \wedge Z)$

Projection/ restriction

Decomposition

Gradual decomposition



$\mathbf{f}^{\intercal} = (1,1,1,1,1,1)$

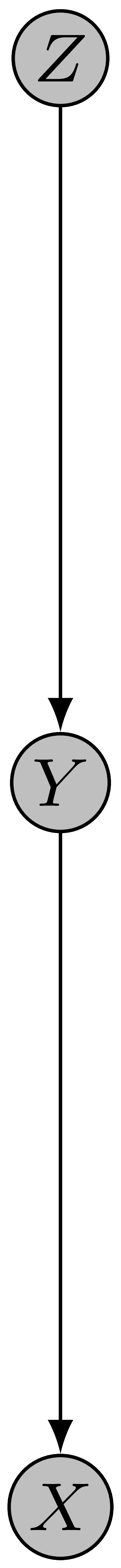
$\{(1,1,1,1,1,1)^{\intercal}\} = D_0$

XY

$(1,1,1,1,1)^{\intercal}$

$\{(1,0,0,1,1)^{\intercal}, (0,1,1,0,0)^{\intercal}\} = D_3$

Independent decomposition



$\mathbf{f}^{\intercal} = (1,1,1,1,1,1)$

XY

$\{(1,1,1,1,1,1)^{\intercal}\} = D_0$

$\mathbf{f}^{\intercal}_{XY} = (1,1,1,1,1)$

$\{(1,0,0,1,1)^{\intercal}, (0,1,1,0,0)^{\intercal}\} = D_3$

Gradual decomposition
Independent decomposition
$(X \wedge Y \wedge Z \,|\top)$
$(X \wedge Y | Z)$
$(X | \, Y \wedge Z)$
Projection/ restriction
Decomposition
$\mathbf{f}^\intercal = (1, 1, 1, 1, 1, 1)$
$\{(1, 1, 1, 1, 1, 1)^\intercal\} = D_0$
XY
restriction of Farkas coefficients to the subsystem XY
$(1, 1, 1, 1, 1)^\intercal$
$\{(1, 0, 0, 1, 1)^\intercal, (0, 1, 1, 0, 0)^\intercal\} = D_3$
$\mathbf{f}^\intercal = (1, 1, 1, 1, 1, 1)$
XY
$\{(1, 1, 1, 1, 1, 1)^\intercal\} = D_0$
$\mathbf{f}^\intercal_{XY} = (1, 1, 1, 1, 1)$
$\{(1, 0, 0, 1, 1)^\intercal, (0, 1, 1, 0, 0)^\intercal\} = D_3$
Z
Y
X
Z
Y
X


Gradual decomposition
Independent decomposition
$(X \wedge Y \wedge Z \mid \top)$
$(X \wedge Y \mid Z)$
$(X \mid Y \wedge Z)$
Projection/ restriction
Decomposition
$\mathbf{f}^\intercal = (1,1,1,1,1,1)$
$\{(1,1,1,1,1,1)^\intercal\} = D_0$
XY
restriction of Farkas coefficients to the subsystem XY
$(1,1,1,1,1)^\intercal$
$\mathbf{f}_{XY}^\intercal = (1,1,1,1,1)$
The decomposition
$\{(1,0,0,1,1)^\intercal, (0,1,1,0,0)^\intercal\} = D_3$
Z
Y
X

Gradual decomposition
$\mathbf{f}^\intercal = (1,1,1,1,1,1)$
$\{(1,1,1,1,1,1)^\intercal\} = D_0$
XY
$(1,1,1,1,1)^\intercal$
$\{(1,0,0,1,1)^\intercal, (0,1,1,0,0)^\intercal\} = D_3$
Z
Y
X
$(X \wedge Y \wedge Z \mid \top)$
$(X \wedge Y \mid Z)$
$(X \mid Y \wedge Z)$
Independent decomposition
Projection/ restriction
Decomposition
$\mathbf{f}^\intercal = (1,1,1,1,1,1)$
XY
$\{(1,1,1,1,1,1)^\intercal\} = D_0$
$\mathbf{f}^\intercal_{XY} = (1,1,1,1,1)$
$\{(1,0,0,1,1)^\intercal, (0,1,1,0,0)^\intercal\} = D_3$
Z
Y
X

Gradual decomposition

$(X \wedge Y \wedge Z \mid \top)$
$(X \wedge Y \mid Z)$
$(X \mid Y \wedge Z)$

Independent decomposition

Projection/ restriction

Decomposition

Z, Y, X

$\mathbf{f}^\intercal = (1,1,1,1,1,1)$

$\{(1,1,1,1,1,1)^\intercal\} = D_0$

XY

$(1,1,1,1,1)^\intercal$

$\{(1,0,0,1,1)^\intercal, (0,1,1,0,0)^\intercal\} = D_3$

X X

$(1,0,0,1)^\intercal \quad (0,1,1,0)^\intercal$

$\{(1,0,0,1)^\intercal, (0,1,1,0)^\intercal\} = D_1$

Z, Y, X

$\mathbf{f}^\intercal = (1,1,1,1,1,1)$

XY

$\{(1,1,1,1,1,1)^\intercal\} = D_0$

$\mathbf{f}^\intercal_{XY} = (1,1,1,1,1)$

$\{(1,0,0,1,1)^\intercal, (0,1,1,0,0)^\intercal\} = D_3$

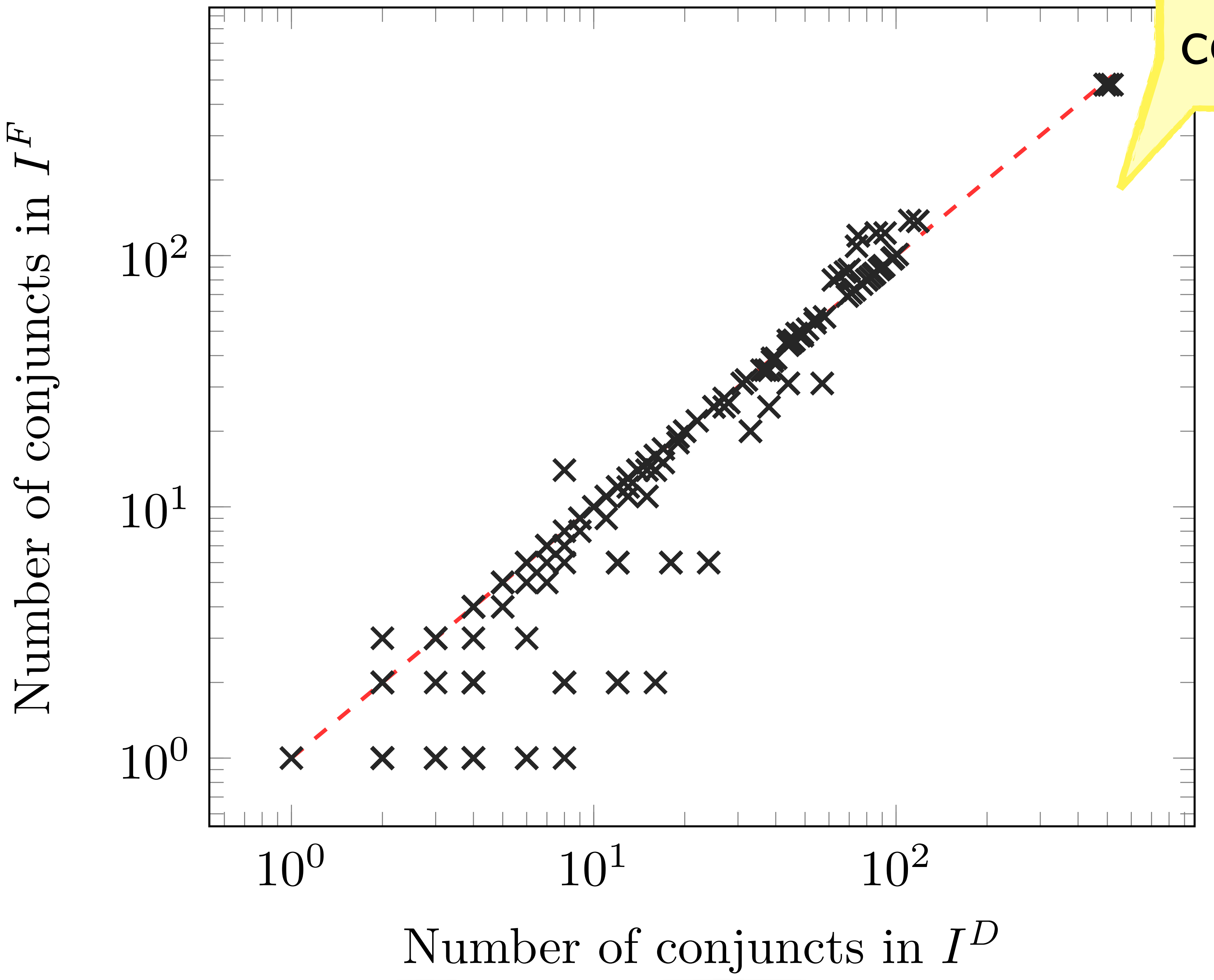

Gradual decomposition
$(X \wedge Y \wedge Z \mid \top)$
$(X \wedge Y \mid Z)$
$(X \mid Y \wedge Z)$
Independent decomposition
Projection/ restriction
Decomposition
$\mathbf{f}^\intercal = (1,1,1,1,1,1)$
$\{(1,1,1,1,1,1)^\intercal\} = D_0$
XY
$(1,1,1,1,1)^\intercal$
$\{(1,0,0,1,1)^\intercal, (0,1,1,0,0)^\intercal\} = D_3$
X
X
$(1,0,0,1)^\intercal$
$(0,1,1,0)^\intercal$
$\{(1,0,0,1)^\intercal, (0,1,1,0)^\intercal\} = D_1$
Z
Y
X
$\mathbf{f}^\intercal = (1,1,1,1,1,1)$
XY
$\{(1,1,1,1,1,1)^\intercal\} = D_0$
$\mathbf{f}^\intercal_{XY} = (1,1,1,1,1)$
X
$\{(1,0,0,1,1)^\intercal, (0,1,1,0,0)^\intercal\} = D_3$
$\mathbf{f}^\intercal_{X} = (1,1,1,1)$
Z
Y
X


Gradual decomposition
Independent decomposition
$(X \wedge Y \wedge Z \mid \top)$
$(X \wedge Y \mid Z)$
$(X \mid Y \wedge Z)$
Projection/ restriction
Decomposition
Z
Y
X
$\mathbf{f}^\intercal = (1,1,1,1,1,1)$
$\{(1,1,1,1,1,1)^\intercal\} = D_0$
XY
$(1,1,1,1,1)^\intercal$
$\{(1,0,0,1,1)^\intercal, (0,1,1,0,0)^\intercal\} = D_3$
X
X
$(1,0,0,1)^\intercal$
$(0,1,1,0)^\intercal$
$\{(1,0,0,1)^\intercal, (0,1,1,0)^\intercal\} = D_1$
$\mathbf{f}^\intercal = (1,1,1,1,1,1)$
XY
$\{(1,1,1,1,1,1)^\intercal\} = D_0$
$\mathbf{f}^\intercal_{XY} = (1,1,1,1,1)$
X
$\{(1,0,0,1,1)^\intercal, (0,1,1,0,0)^\intercal\} = D_3$
$\mathbf{f}^\intercal_{X} = (1,1,1,1)$
$\{(1,1,0,0)^\intercal, (0,0,1,1)^\intercal\} = D_2$

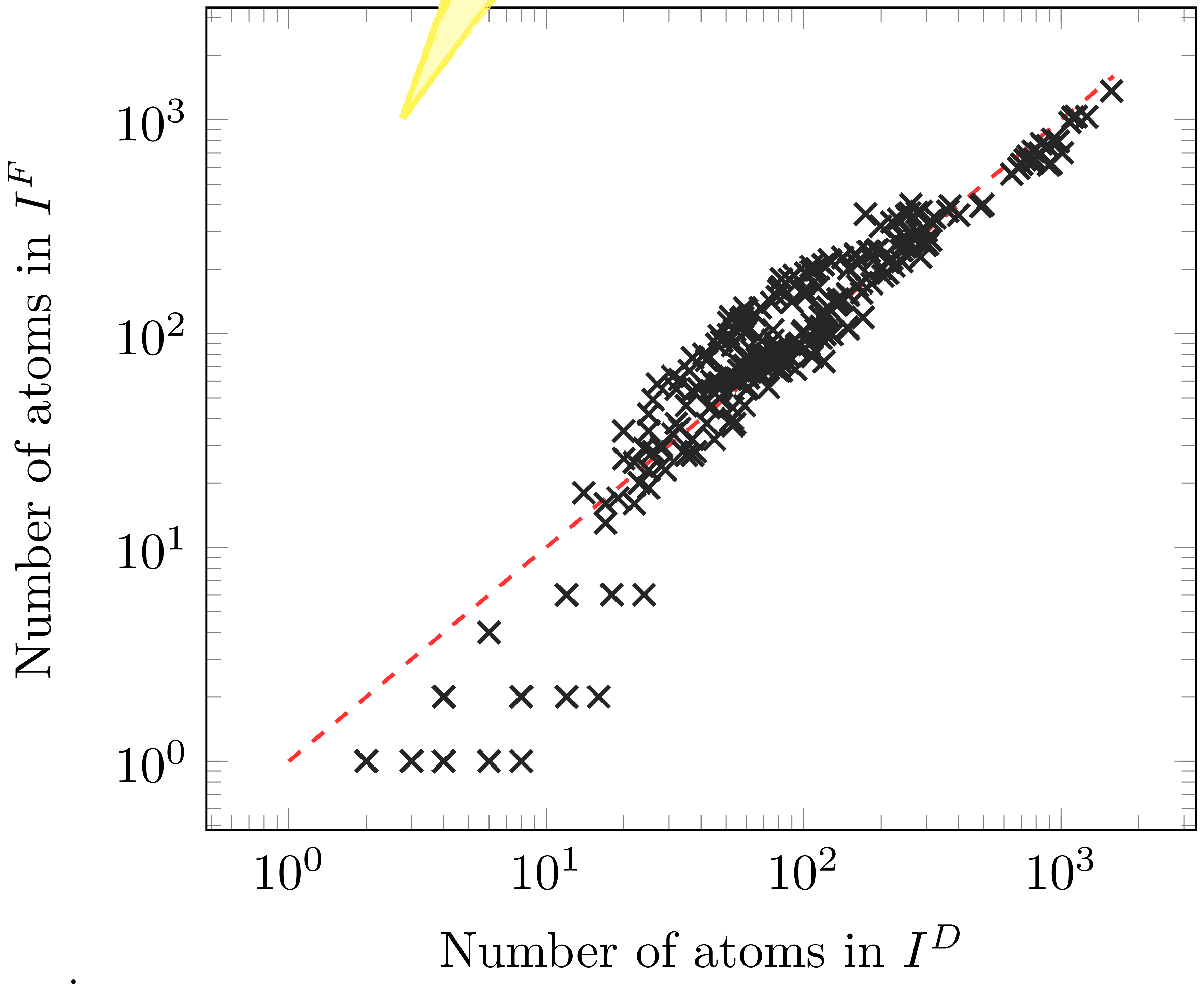

Gradual decomposition
$(X| Y \wedge Z)$
$(X \wedge Y | Z)$
$(X \wedge Y \wedge Z | \top)$
Independent decomposition
Projection/ restriction
Decomposition
$\mathbf{f}^\intercal = (1,1,1,1,1,1)$
$\{(1,1,1,1,1,1)^\intercal\} = D_0$
XY
$(1,1,1,1,1)^\intercal$
$\{(1,0,0,1,1)^\intercal, (0,1,1,0,0)^\intercal\} = D_3$
X
X
$(1,0,0,1)^\intercal$
$(0,1,1,0)^\intercal$
$\{(1,0,0,1)^\intercal, (0,1,1,0)^\intercal\} = D_1$
D1 agrees with D3
$\mathbf{f}^\intercal = (1,1,1,1,1,1)$
XY
$\{(1,1,1,1,1,1)^\intercal\} = D_0$
$\mathbf{f}_{XY}^\intercal = (1,1,1,1,1)$
X
$\{(1,0,0,1,1)^\intercal, (0,1,1,0,0)^\intercal\} = D_3$
$\mathbf{f}_X^\intercal = (1,1,1,1)$
D2 does not agree with D3
$\{(1,1,0,0)^\intercal, (0,0,1,1)^\intercal\} = D_2$
Z
Y
X
Z
Y
X

Experiments:
Farkas (I^F) vs. Decomposed Farkas (I^D) tree interpolants

Benchmarks: 514 QF_LRA UNSAT formulas, up to 12k LoC



91% of benchmarks I^D have strictly more conjuncts



63% of benchmarks I^F have fewer atoms

In practice interpolation algorithms are substantially different and complementary!

Related Work

Proof tree preserving tree interpolation [Christ et al, JAR 2016]

Tree interpolation in Vampire [Blanc et al, LPAR 2013]

Disjunctive interpolants for Horn-clause verification [Rummer et al, CAV 2013]

Strategy synthesis for linear arithmetic games [Farzan et al, POPL 2018]

The ELDARICA Horn Solver [Hojjat et al, FMCAD 2018]

Solving recursion-free Horn clauses [Gupta et al, APLAS 2011]

Applications:

Incremental Verification by SMT-based Summary Repair [Asadi et al, FMCAD 2020]

eVolCheck: Incremental Upgrade Checker for C [Fedyukovich et al, TACAS 2013]

Conclusion

- Investigated the necessary conditions for the Tree Interpolation Property for five state-of-the-art LRA interpolation algorithms.

Interpolation algorithm	TIP
Farkas	✓
Dual Farkas	×
Decomposing Farkas	✓
Dual Decomposing Farkas	×
Flexible Farkas	×

- Lifted a recently introduced approach producing conjunctive LRA interpolants to tree interpolation by introducing gradual decomposition.

- Opening the possibility of using different proof-based interpolation portfolios in applications that require TIP.

Conclusion

- Investigated the necessary conditions for the Tree Interpolation Property for five state-of-the-art LRA interpolation algorithms.

Interpolation algorithm	TIP
Farkas	✓
Dual Farkas	×
Decomposing Farkas	✓
Dual Decomposing Farkas	×
Flexible Farkas	×

- Lifted a recently introduced approach producing conjunctive LRA interpolants to tree interpolation by introducing gradual decomposition.

- Opening the possibility of using different proof-based interpolation portfolios in applications that require TIP.

http://verify.inf.usi.ch/upprover

Conclusion

- Investigated the necessary conditions for the Tree Interpolation Property for five state-of-the-art LRA interpolation algorithms.

Interpolation algorithm	TIP
Farkas	✓
Dual Farkas	×
Decomposing Farkas	✓
Dual Decomposing Farkas	×
Flexible Farkas	×

- Lifted a recently introduced approach producing conjunctive LRA interpolants to tree interpolation by introducing gradual decomposition.

- Opening the possibility of using different proof-based interpolation portfolios in applications that require TIP.

Thanks for your attention!

http://verify.inf.usi.ch/upprover