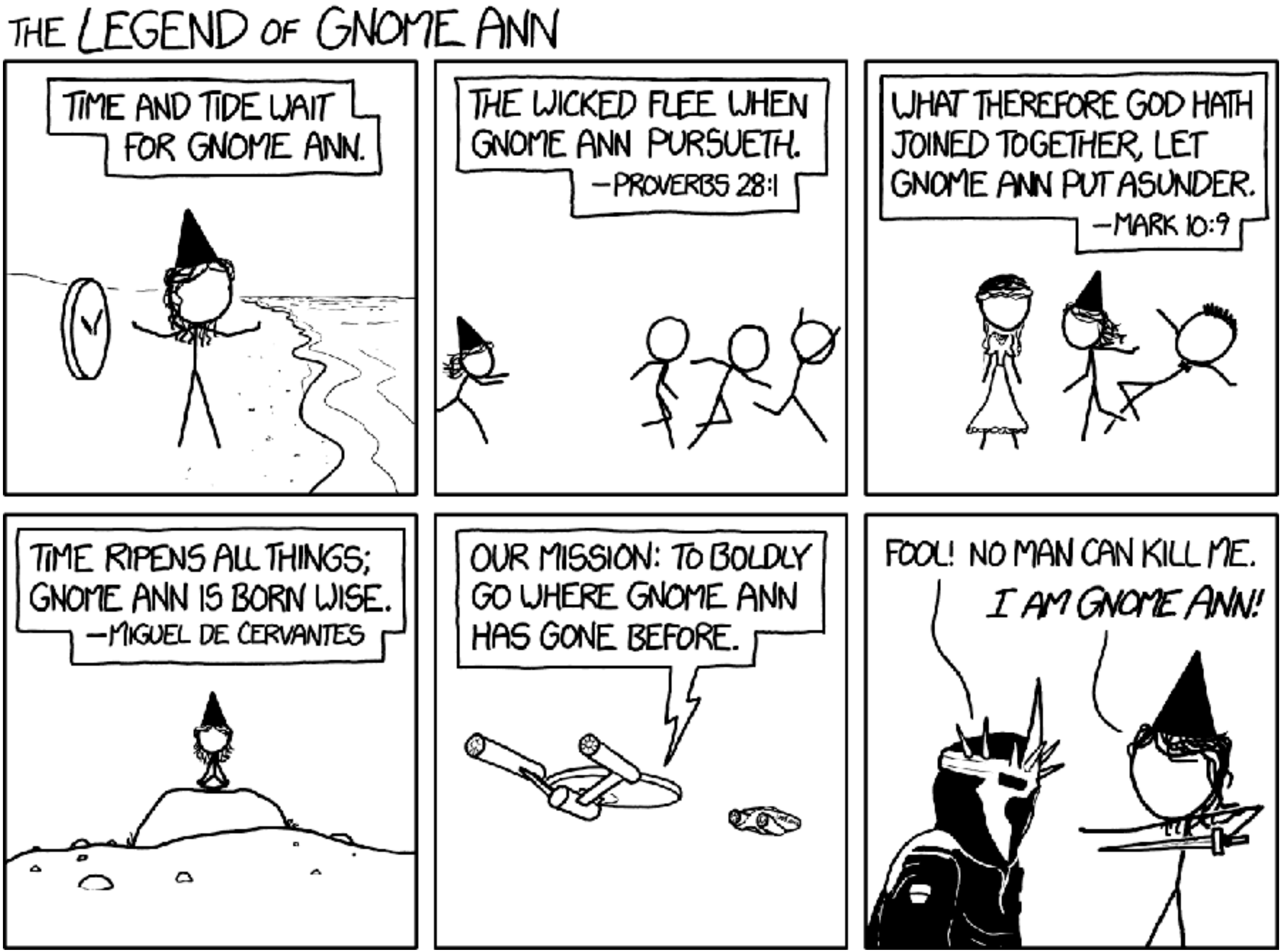
# LRA Interpolants from No Man's Land

Leonardo Alt, **Antti E. J. Hyvärinen**, and Natasha Sharygina
University of Lugano, Switzerland


THE LEGEND OF GNOME ANN
TIME AND TIDE WAIT FOR GNOME ANN.
THE WICKED FLEE WHEN GNOME ANN PURSUETH.
—PROVERBS 28:1
WHAT THEREFORE GOD HATH JOINED TOGETHER, LET GNOME ANN PUT ASUNDER.
—MARK 10:9
TIME RIPENS ALL THINGS; GNOME ANN IS BORN WISE.
—MIGUEL DE CERVANTES
OUR MISSION: TO BOLDLY GO WHERE GNOME ANN HAS GONE BEFORE.
FOOL! NO MAN CAN KILL ME.
I AM GNOME ANN!

# Motivation

The goal: Finding the right proof

The tool: Make interpolation on LRA more flexible

The application: LRA for abstractions in software model checking

The keywords: SMT solving, function summaries, labeled interpolation systems

# Interpolants

Given two formulas $\mathrm{A}$ and $\mathrm{B}$ such that

$$A \wedge B \rightarrow \bot$$

an interpolant is a formula $I$ such that

$$\mathrm{Vars}(I) \subseteq \mathrm{Vars}(A) \cap \mathrm{Vars}(B)$$

$$A \rightarrow I$$

$$I \wedge B \rightarrow \bot$$

# Interpolants

Given two formulas $\mathrm{A}$ and $\mathrm{B}$ such that

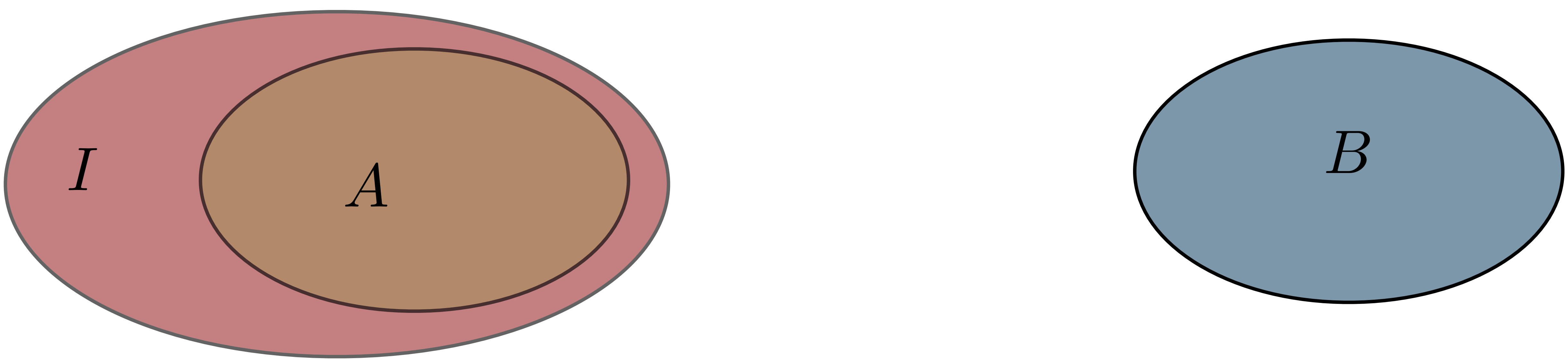
$$A \wedge B \rightarrow \bot$$

an interpolant is a formula $I$ such that

$$\mathrm{Vars}(I) \subseteq \mathrm{Vars}(A) \cap \mathrm{Vars}(B)$$

$$A \rightarrow I$$

$$I \wedge B \rightarrow \bot$$



# Interpolants

Given two formulas $\mathrm{A}$ and $\mathrm{B}$ such that

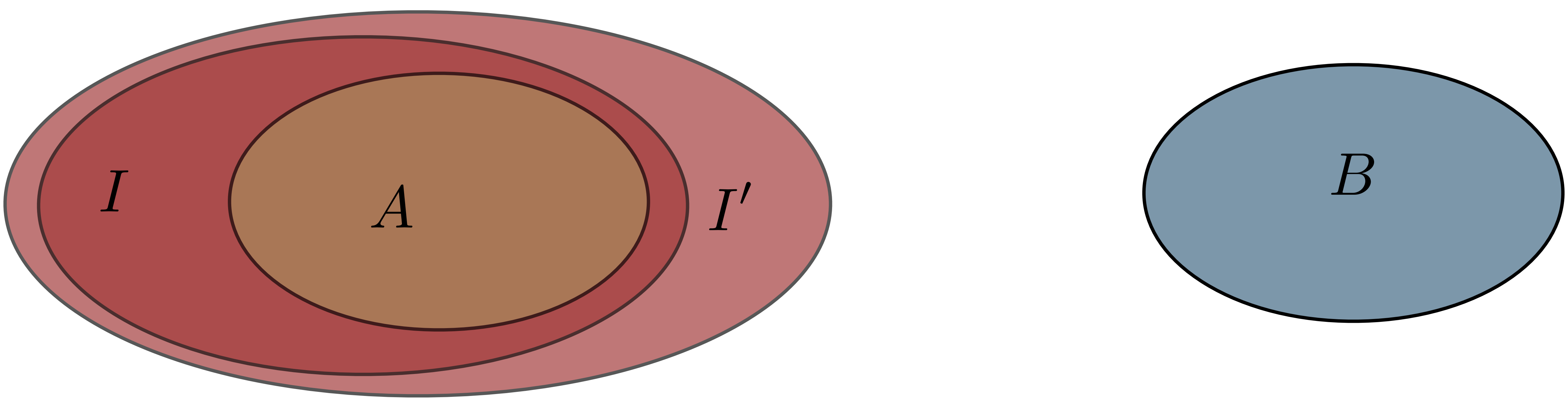
$$A \wedge B \to \bot$$

an interpolant is a formula $I$ such that

$$\mathrm{Vars}(I) \subseteq \mathrm{Vars}(A) \cap \mathrm{Vars}(B)$$

$$A \to I$$

$$I \wedge B \to \bot$$



# Interpolants

Given two formulas $\mathrm{A}$ and $\mathrm{B}$ such that

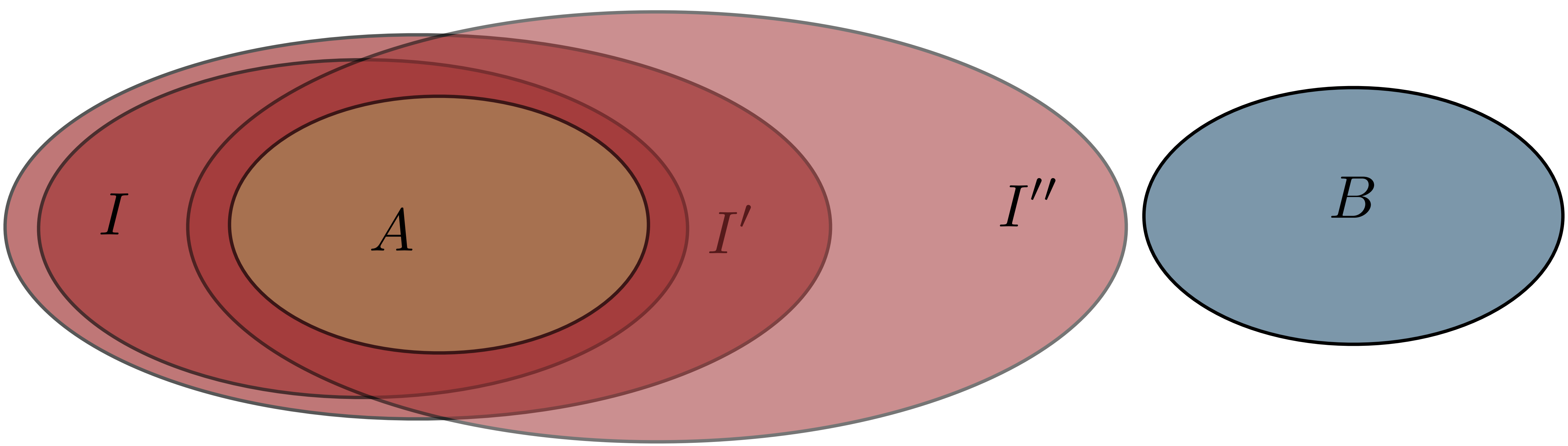
$$A \wedge B \to \bot$$

an interpolant is a formula $I$ such that

$$\mathrm{Vars}(I) \subseteq \mathrm{Vars}(A) \cap \mathrm{Vars}(B)$$

$$A \to I$$

$$I \wedge B \to \bot$$



# Interpolation in Proofs

1. Find a concrete proof for a simple case

2. Generalise the proof

3. Try to prove the general case

# Interpolation in Proofs

1. Find a concrete proof for a simple case

2. Generalise the proof

3. Try to prove the general case

$$\underbrace{S(x_0)}_{\text{A}} \wedge \underbrace{T(x_0, x_1) \wedge \ldots \wedge T(x_{k-1}, x_k) \wedge Err(x_k)}_{\text{B}}$$

# Interpolation in Proofs

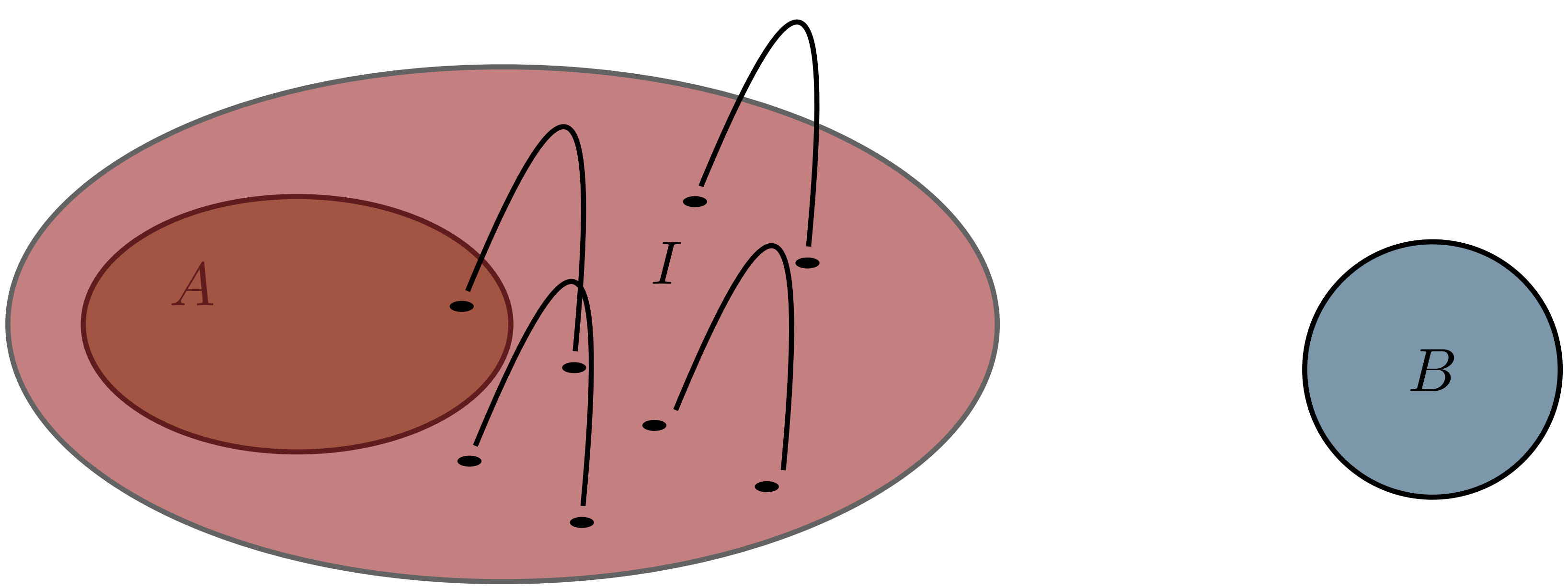
1. Find a concrete proof for a simple case

2. Generalise the proof

3. Try to prove the general case

$$\underbrace{S(x_0)}_{\text{A}} \wedge \underbrace{T(x_0, x_1) \wedge \ldots \wedge T(x_{k-1}, x_k) \wedge Err(x_k)}_{\text{B}}$$

$$I(x) \wedge T(x, x') \rightarrow I(x')$$



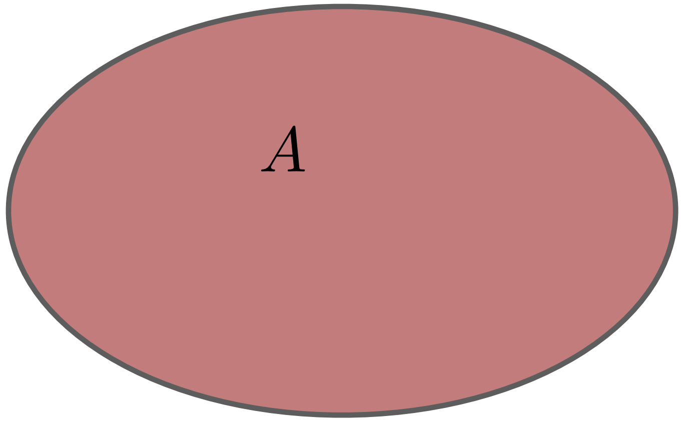
# Example: HiFrog

Given a C program and a set of assertions

# Example: HiFrog

Given a C program and a set of assertions

1. Construct a BMC instance of the program



# Example: HiFrog

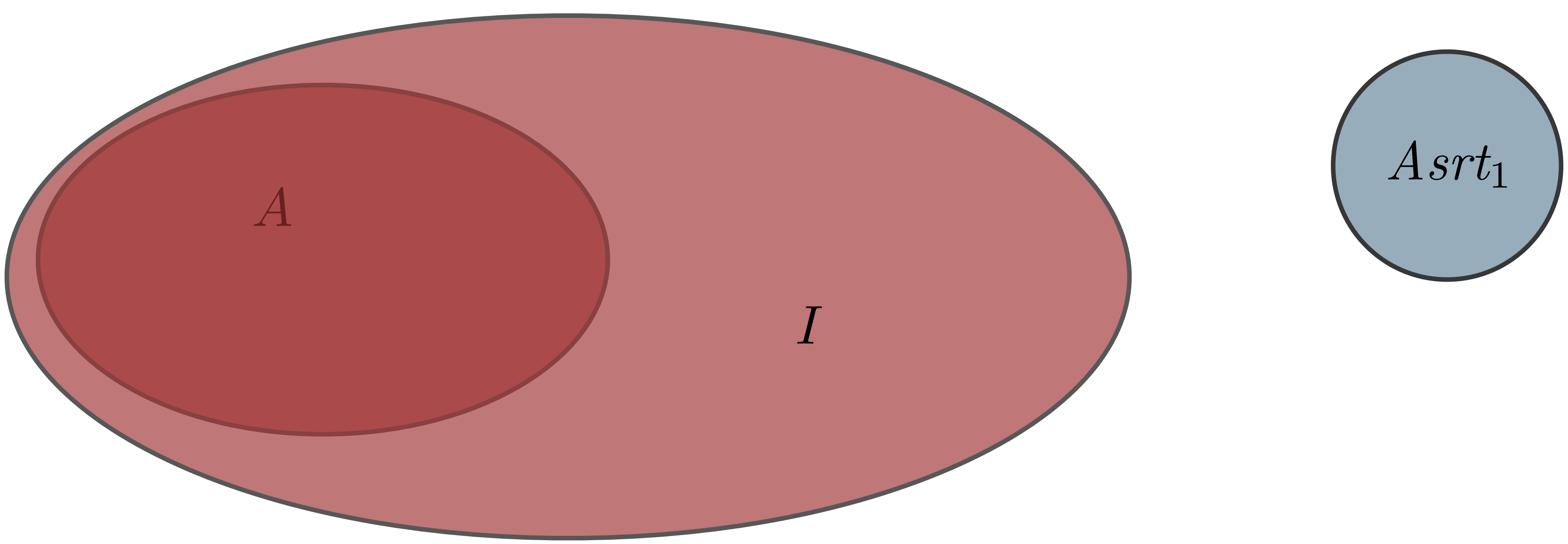
Given a C program and a set of assertions

1. Construct a BMC instance of the program
2. Check the first assertion against the BMC instance

# Example: HiFrog

Given a C program and a set of assertions

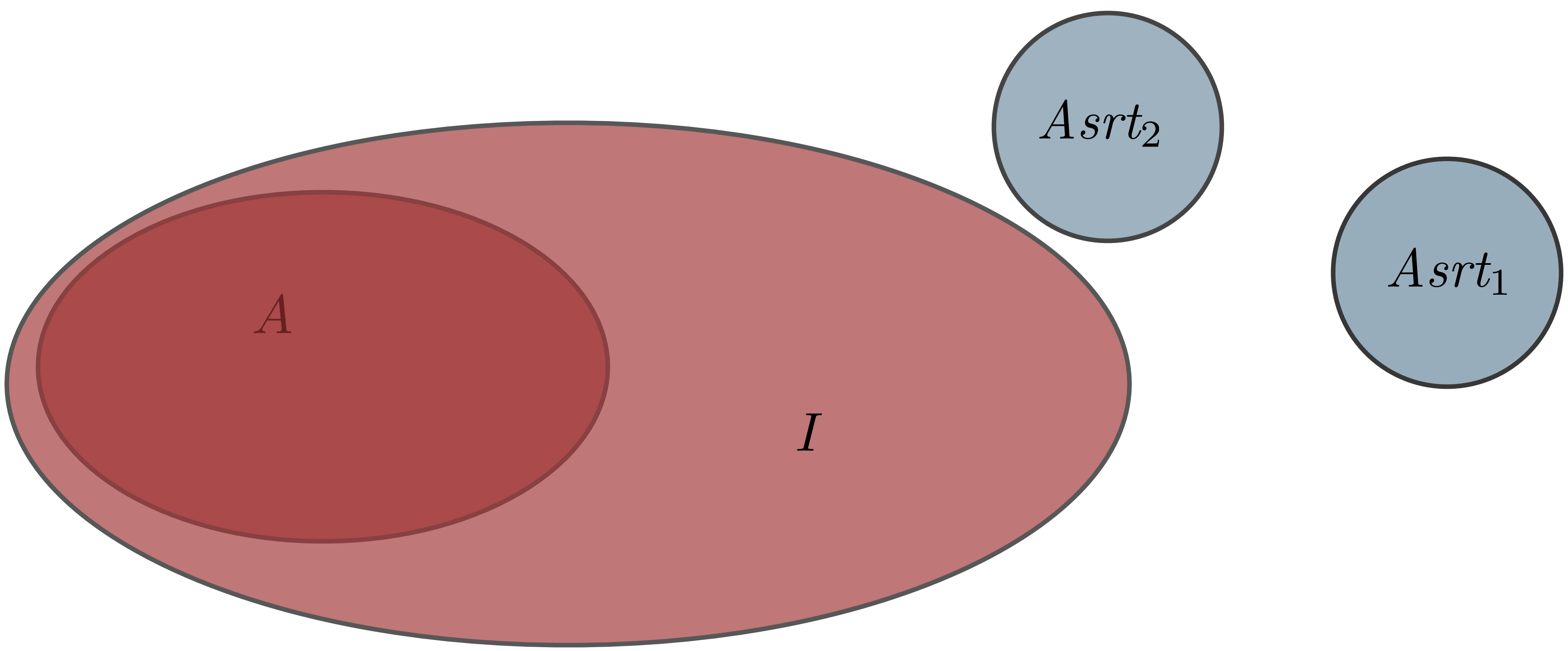
1. Construct a BMC instance of the program
2. Check the first assertion against the BMC instance
3. Compute an interpolant out of the proof



# Example: HiFrog

Given a C program and a set of assertions

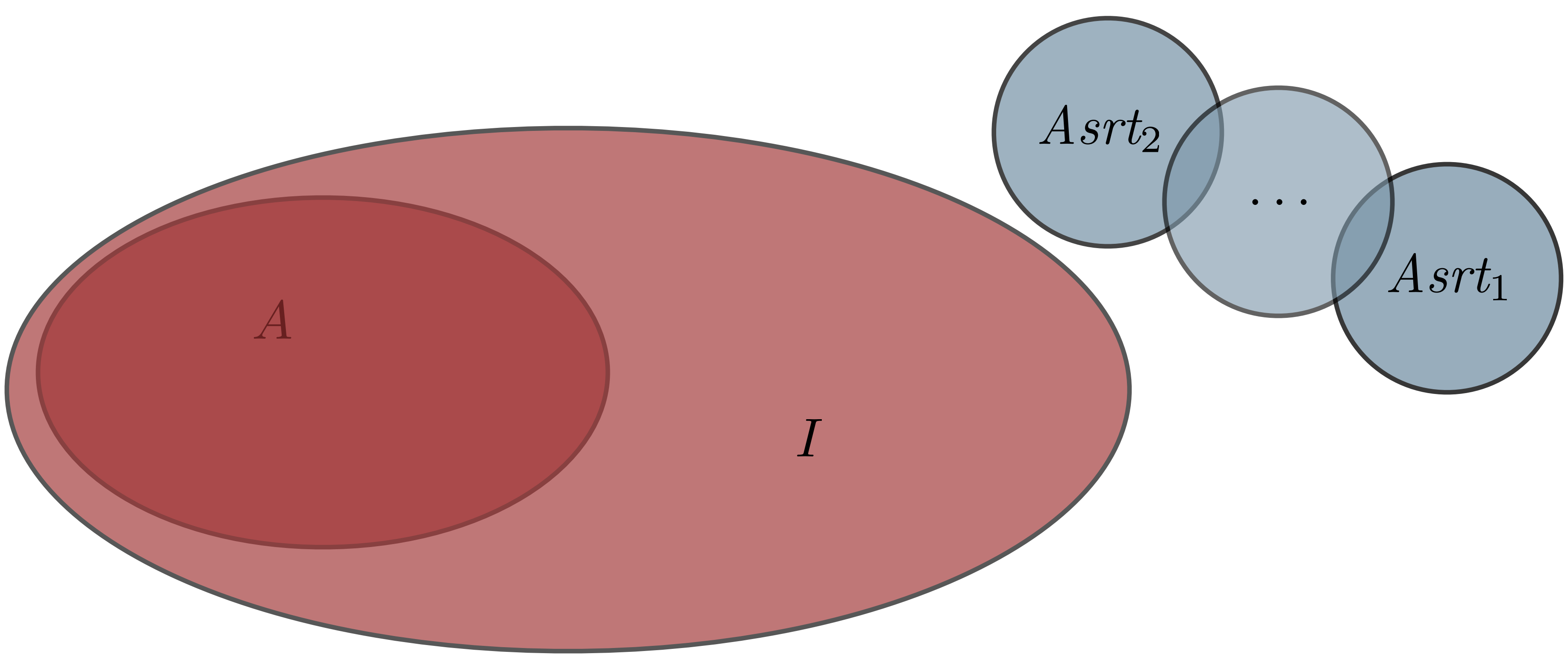
1. Construct a BMC instance of the program
2. Check the first assertion against the BMC instance
3. Compute an interpolant out of the proof
4. Use the interpolant for checking the consequent assertions



# Example: HiFrog

Given a C program and a set of assertions

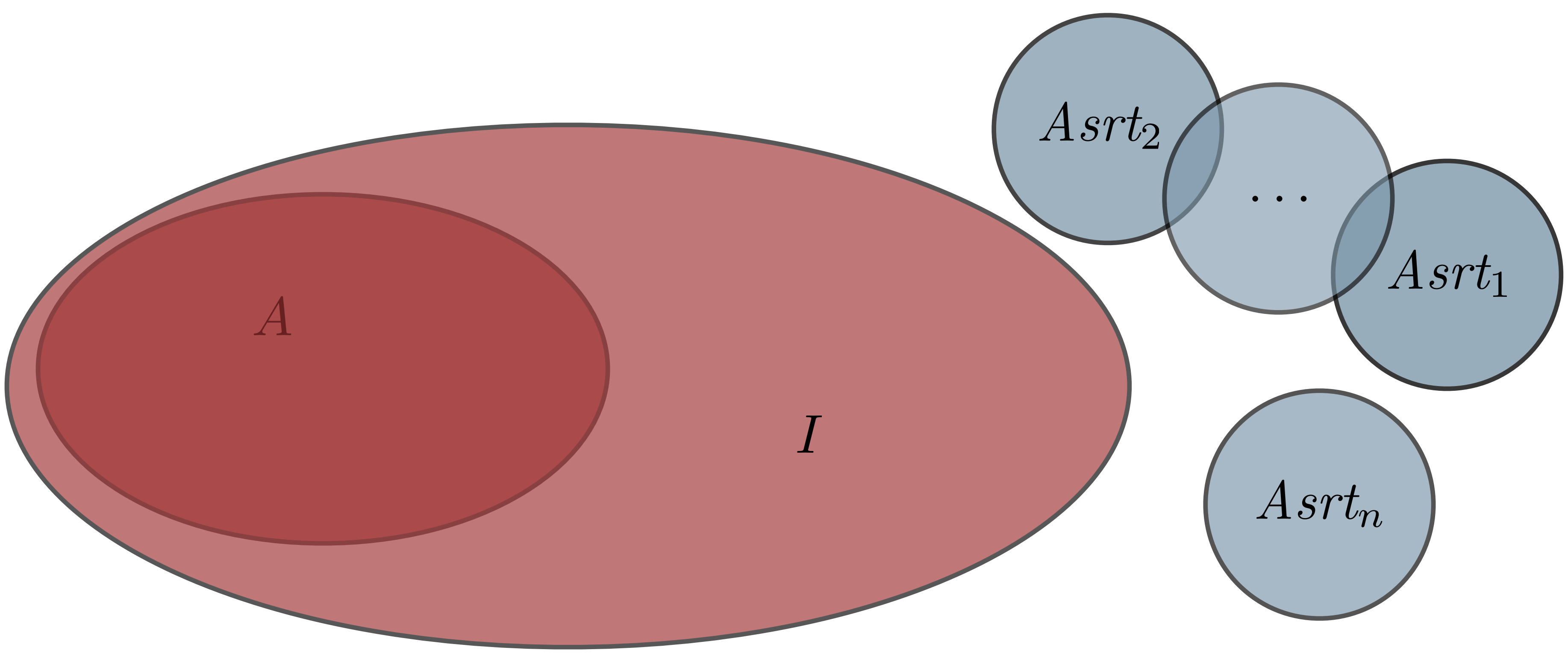
1. Construct a BMC instance of the program
2. Check the first assertion against the BMC instance
3. Compute an interpolant out of the proof
4. Use the interpolant for checking the consequent assertions



# Example: HiFrog

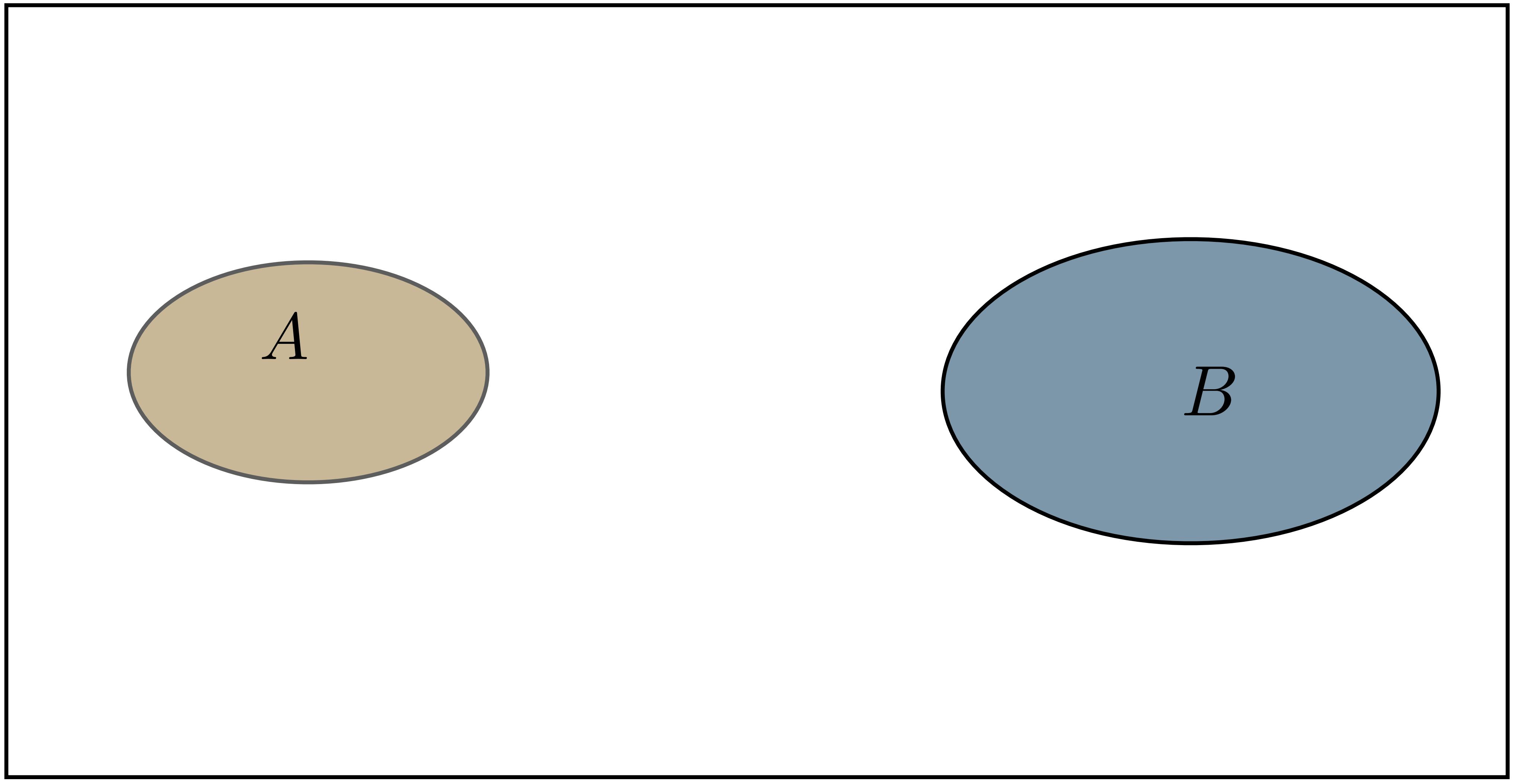
Given a C program and a set of assertions

1. Construct a BMC instance of the program
2. Check the first assertion against the BMC instance
3. Compute an interpolant out of the proof
4. Use the interpolant for checking the consequent assertions

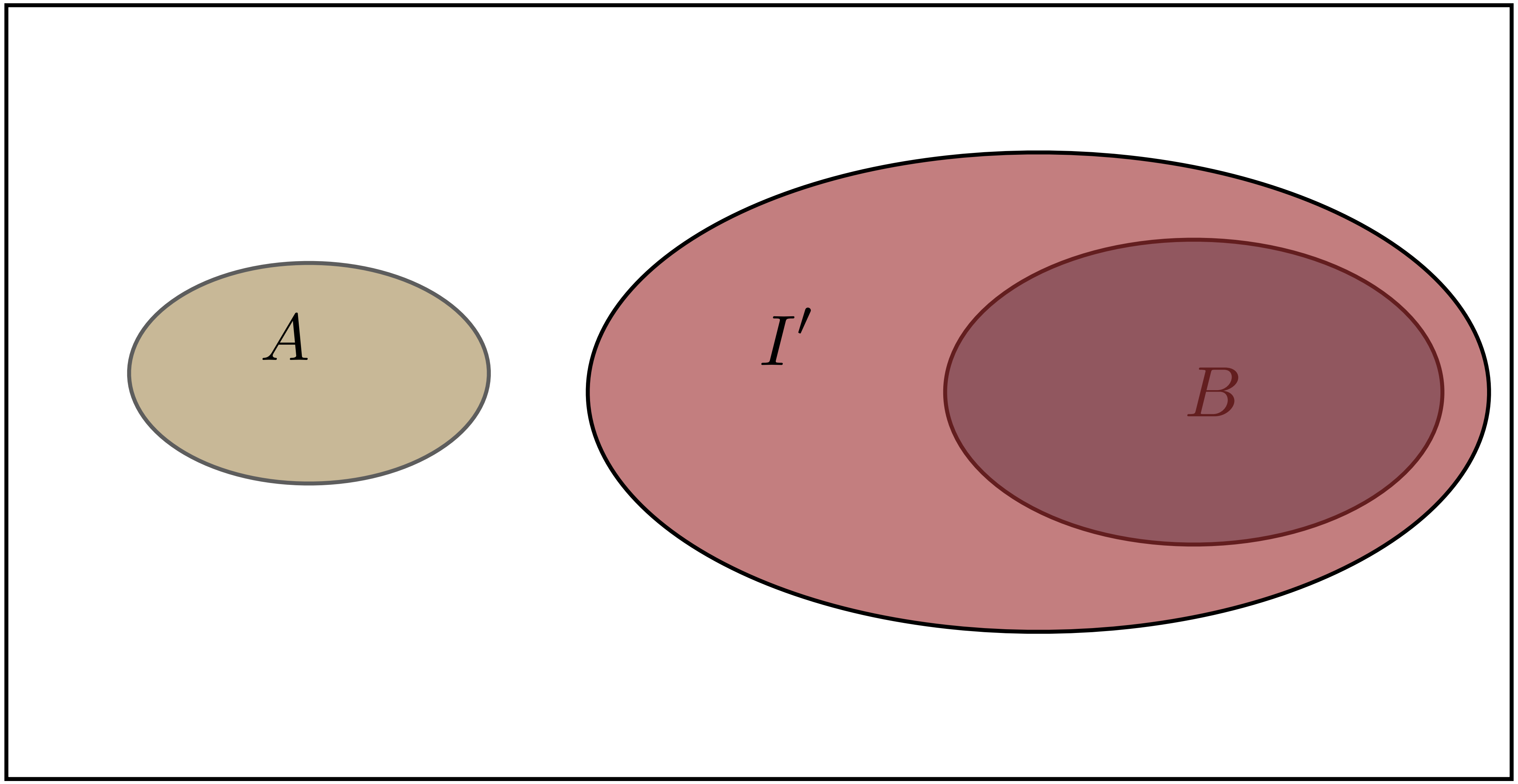


# Duality of Interpolants

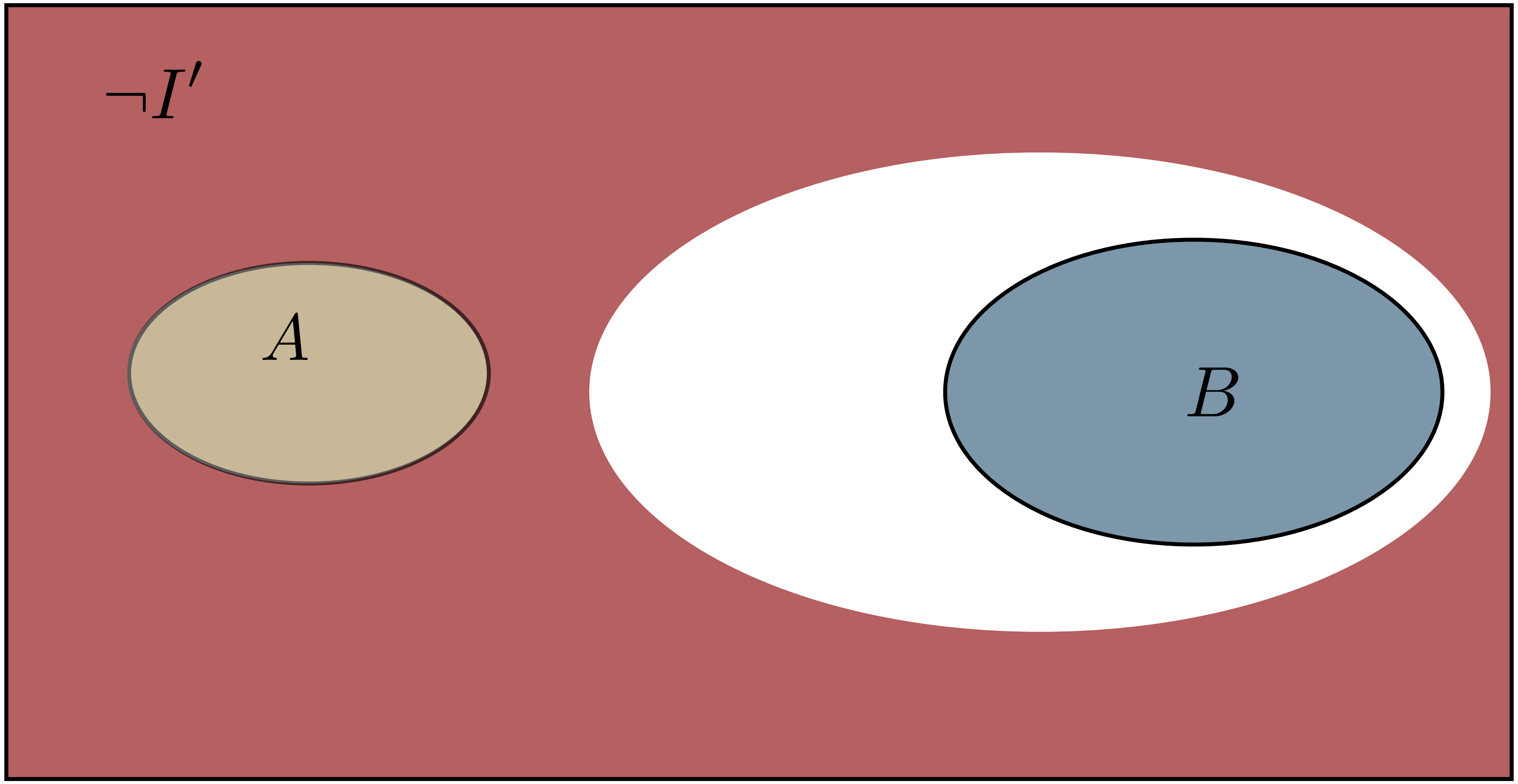
# Duality of Interpolants



# Duality of Interpolants



# Duality of Interpolants



# What is LRA

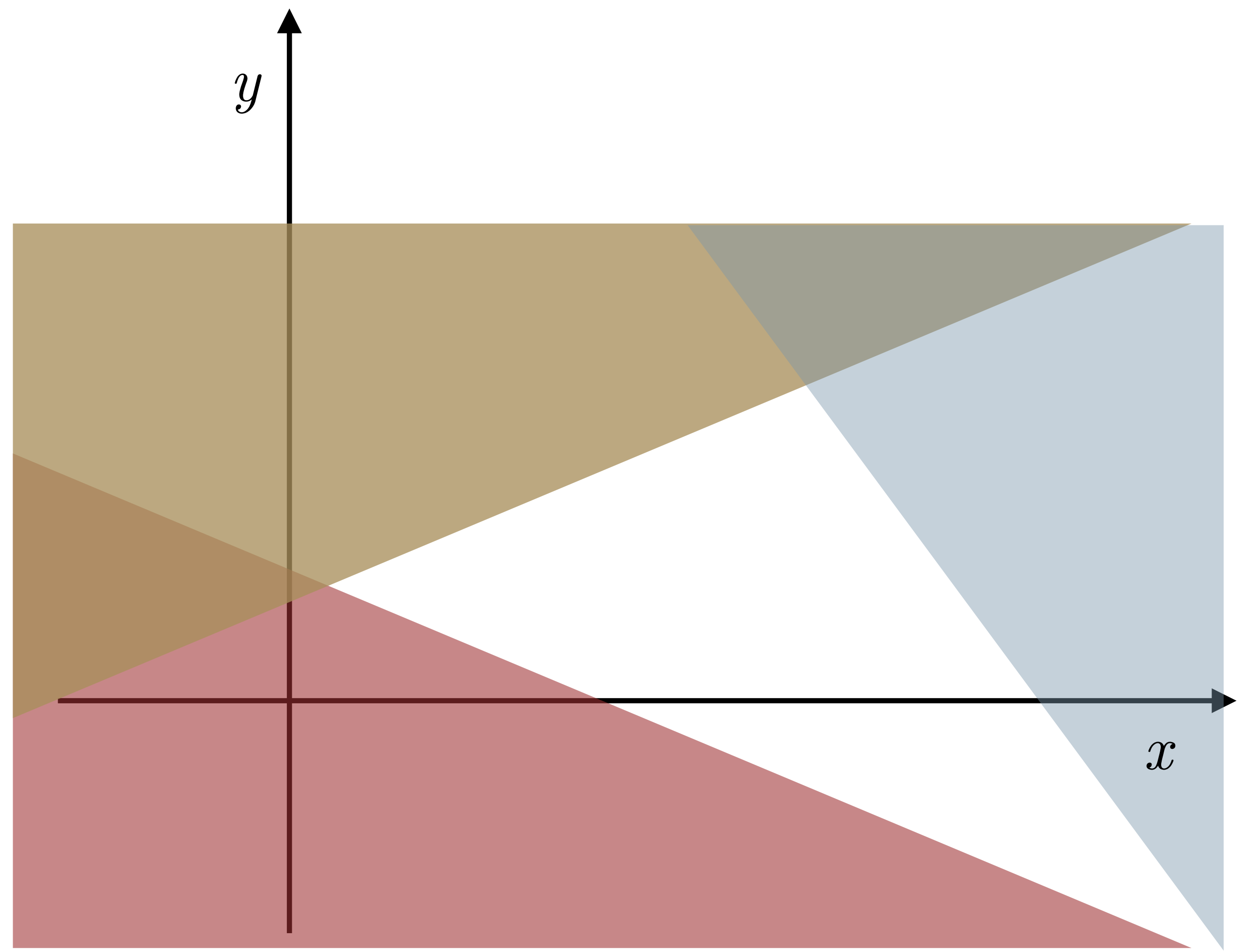
Given a set of linear inequalities over real-valued variables,

determine if there are values for the variables that satisfy all the inequalities

$$0 \leq 0.5x + 3y - 2z$$

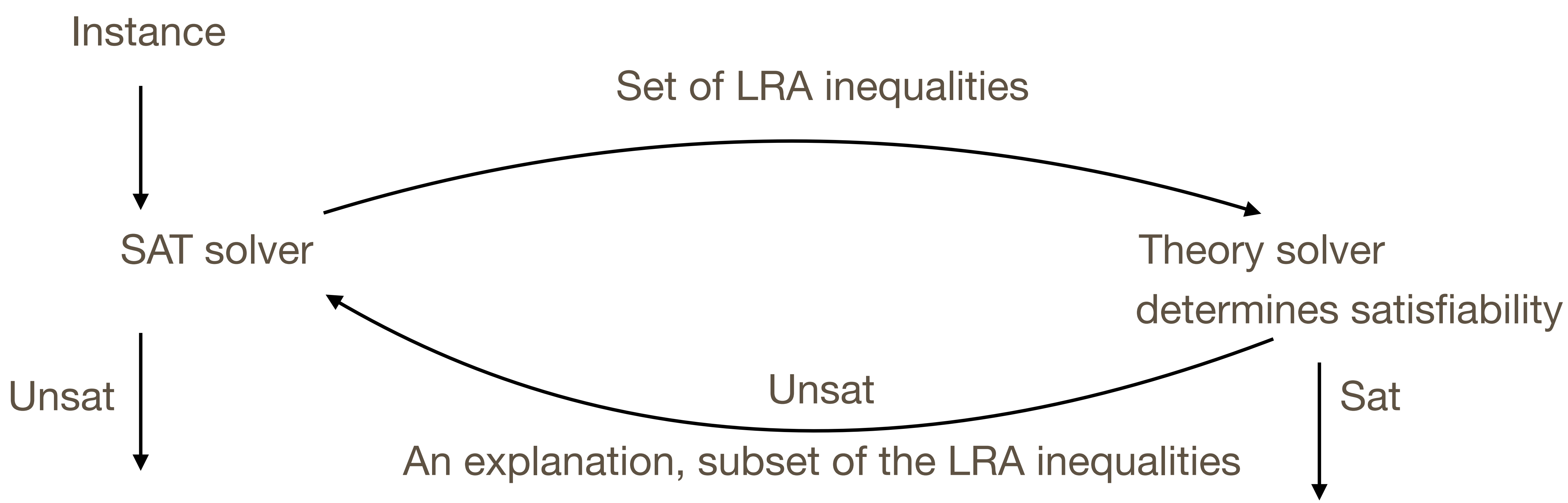
$$5 > y$$

$$\cancel{x^2 + 2xy + y^2 > 1}$$




In 2 dimensions: determine whether half planes have a non-empty intersection

# Solving LRA in SMT




The theory solver for LRA is based on the Simplex algorithm

# Simplex in SMT

A pre-processing step:

- All inequalities are written so that left side is a constant and right side a linear expression

We end up with two types of entities:

- Bounds on variables
- Bounds on sums of the variables

The idea is to repeatedly adjust variable values to satisfy bounds on the sums, and change the role of the variables and the sums.

$$0 \leq x + 6y - 4z$$

$$0.3 \geq x + 2y$$

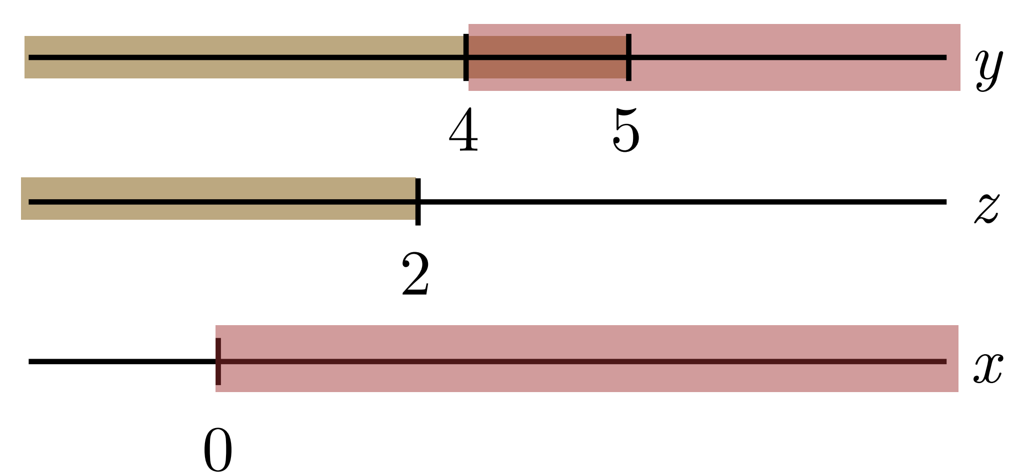
$$1 < x - 2y + 3z$$

$$5 > y$$

$$4 < y$$

$$2 > z$$

$$0 \geq x$$

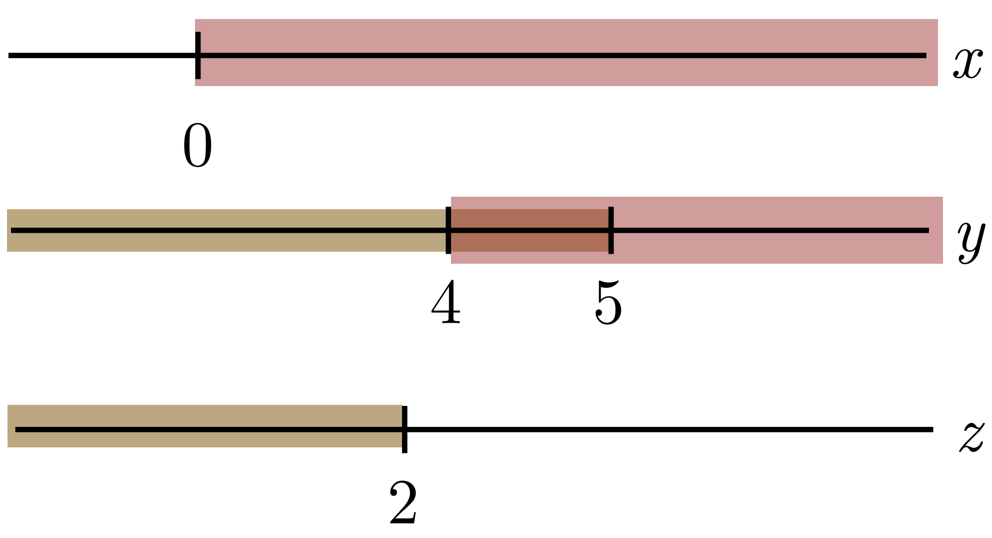


# Simplex Example

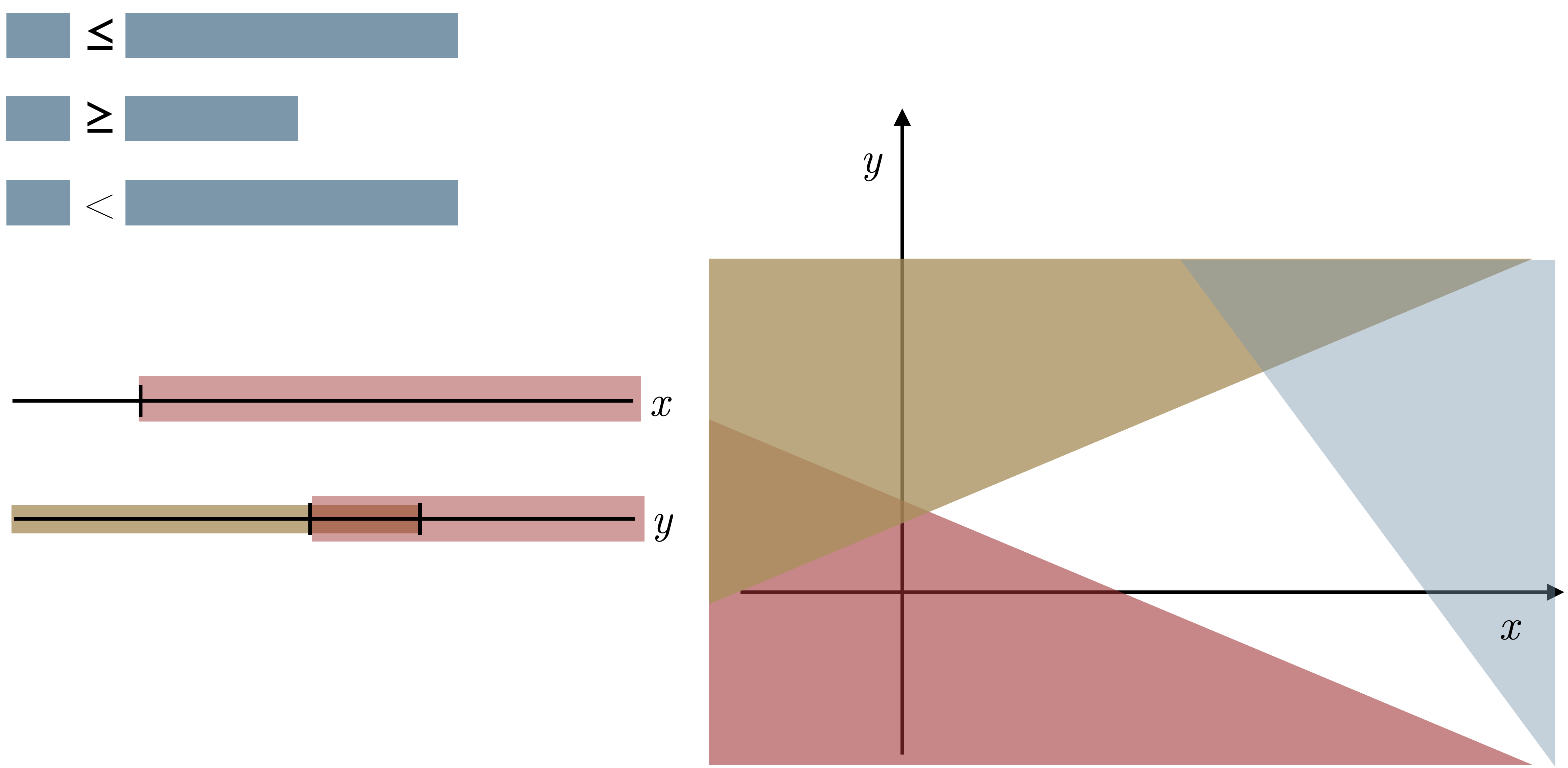
$$0 \quad \leq 0.5x + 3y - 2z$$

$$0.3 \geq x + 2y$$

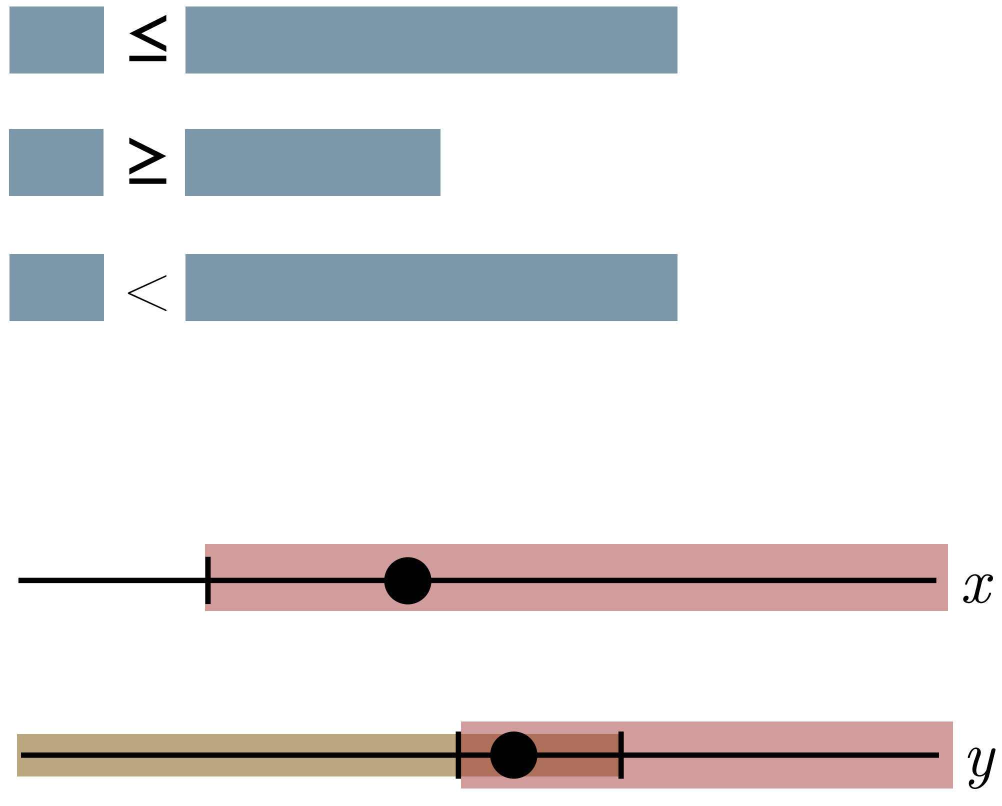
$$1 \quad < x - 2y + 3z$$

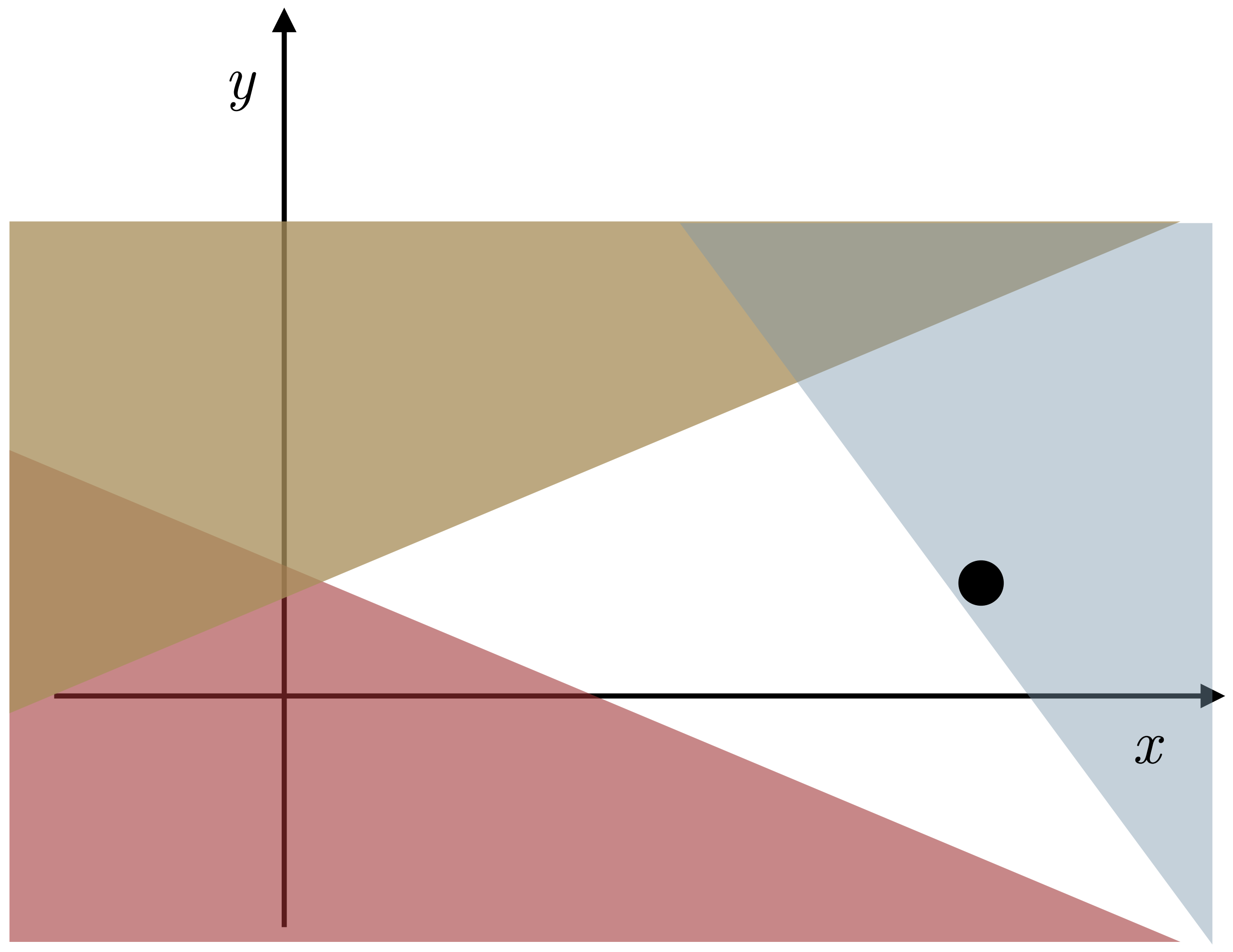


# Simplex Example

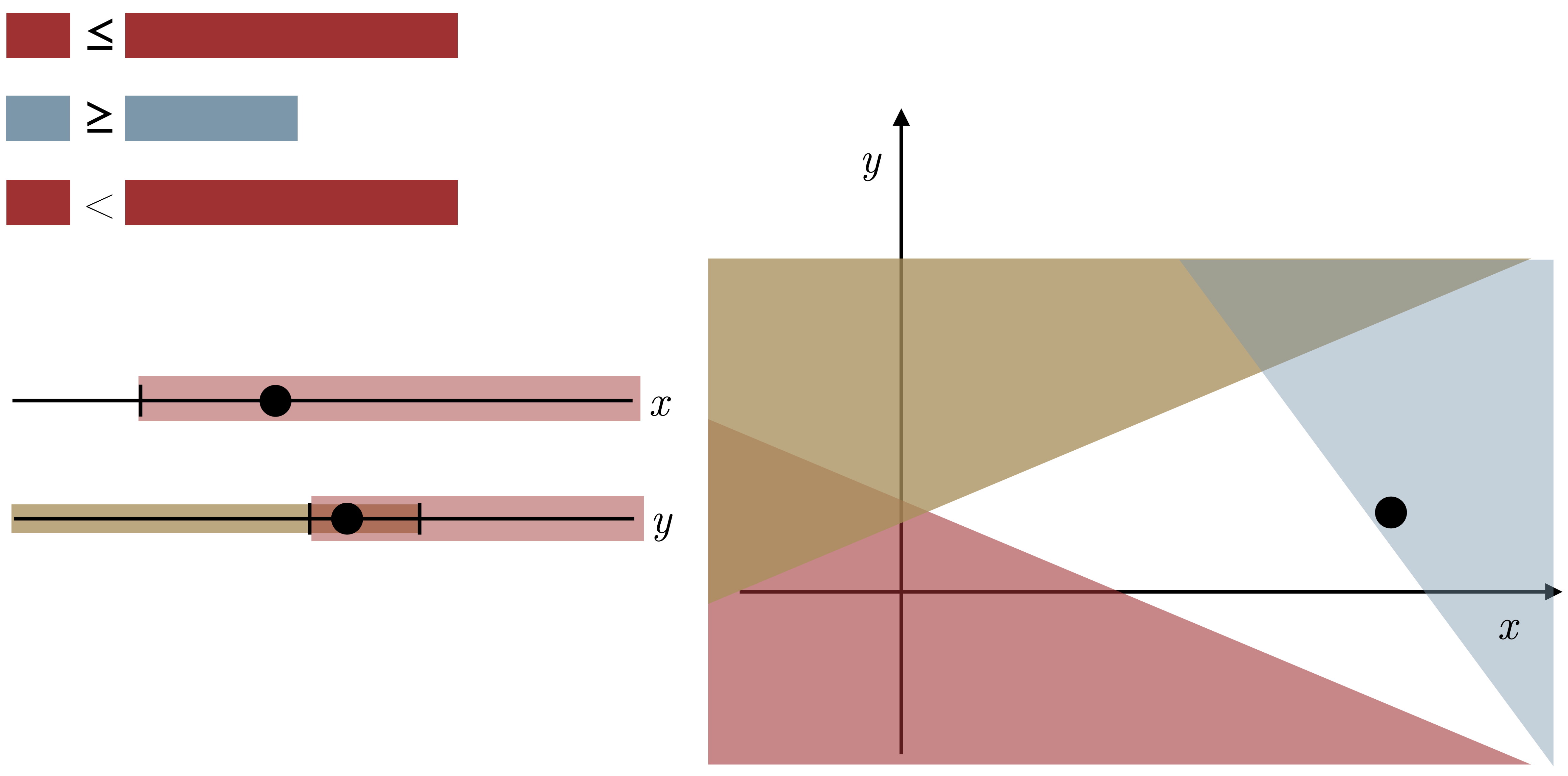


# Simplex Example

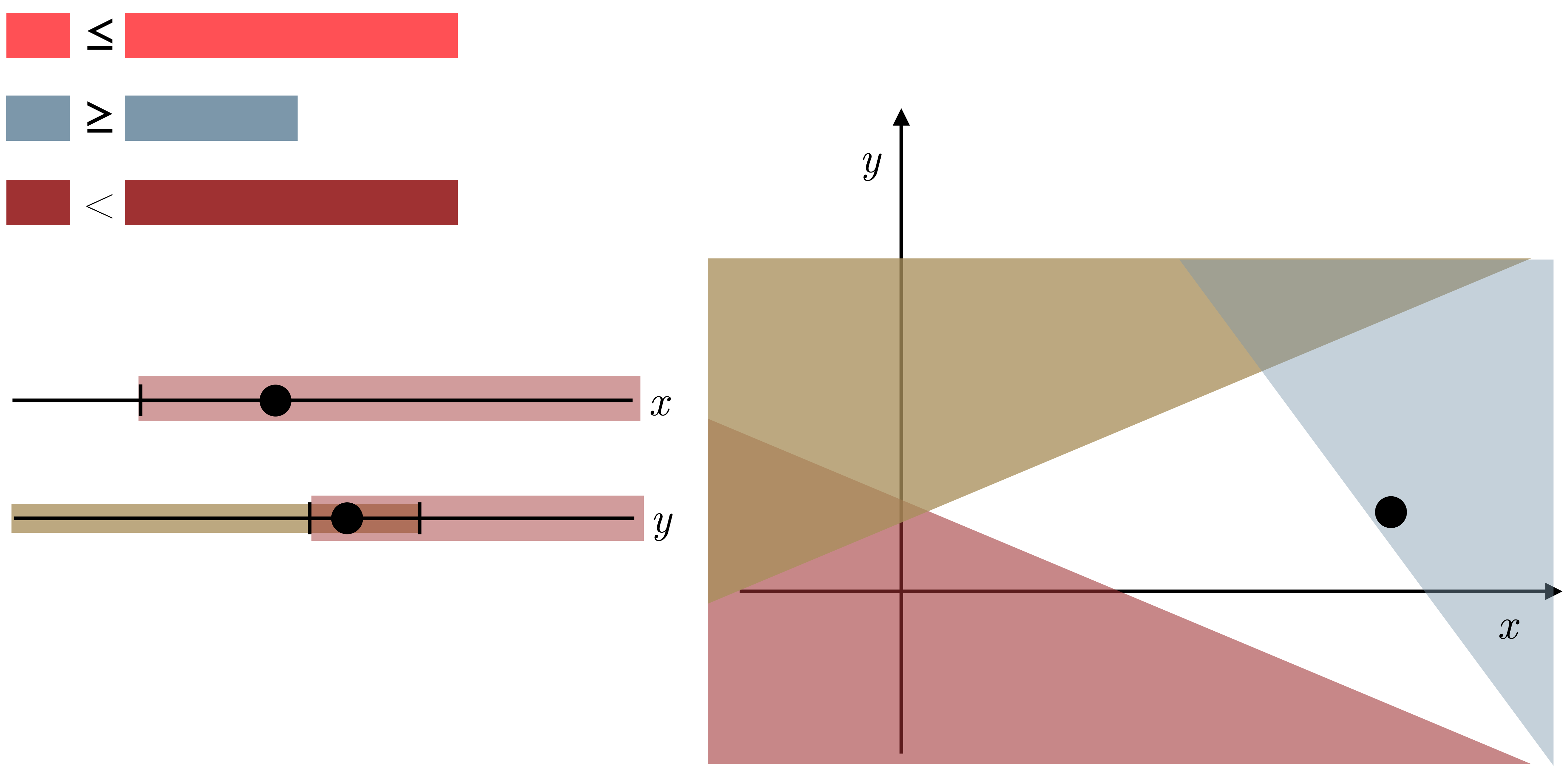


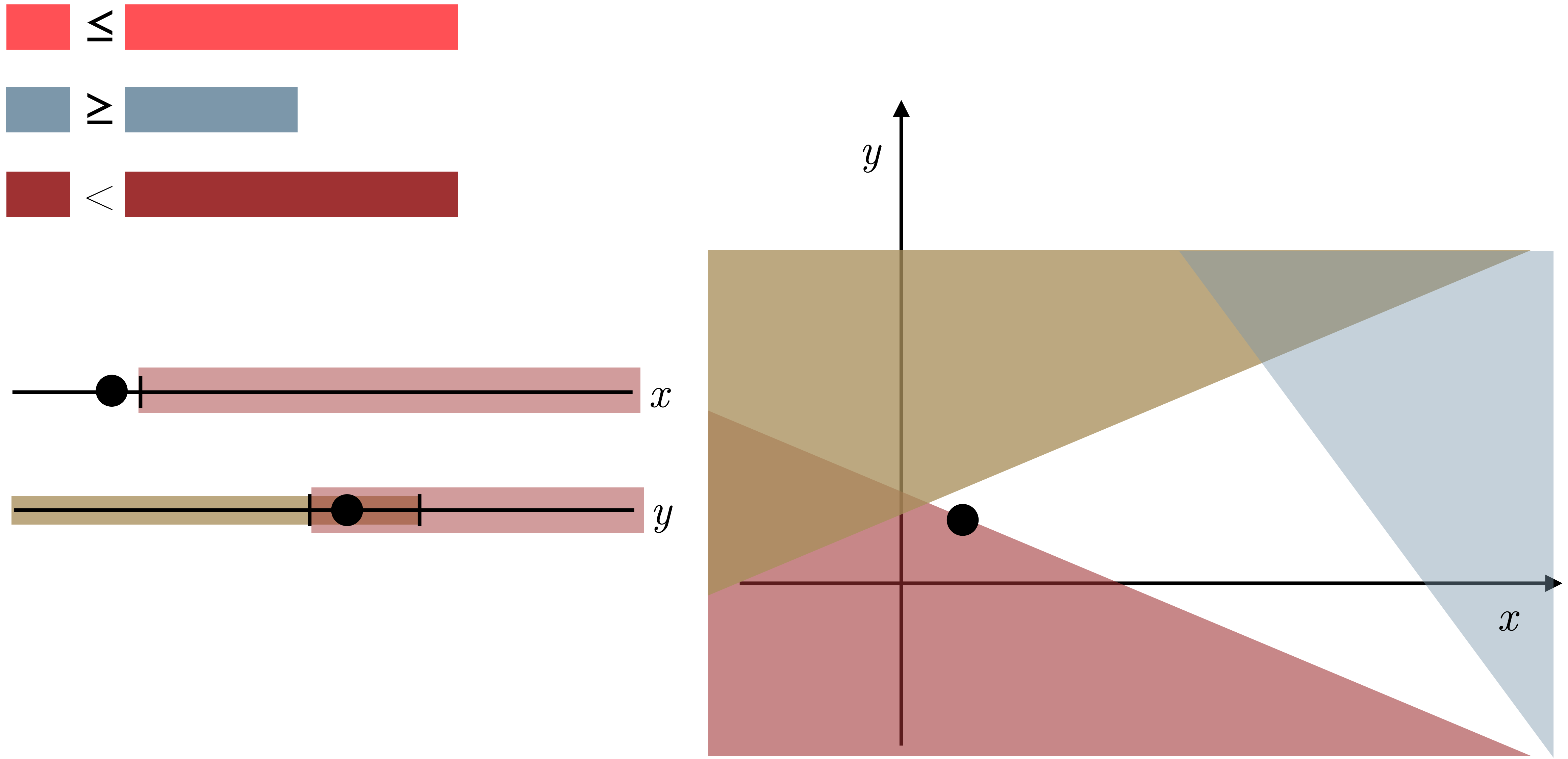
# Simplex Example



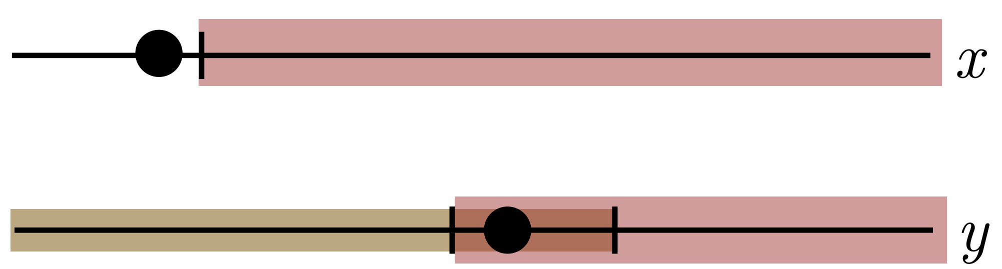
# Simplex Example

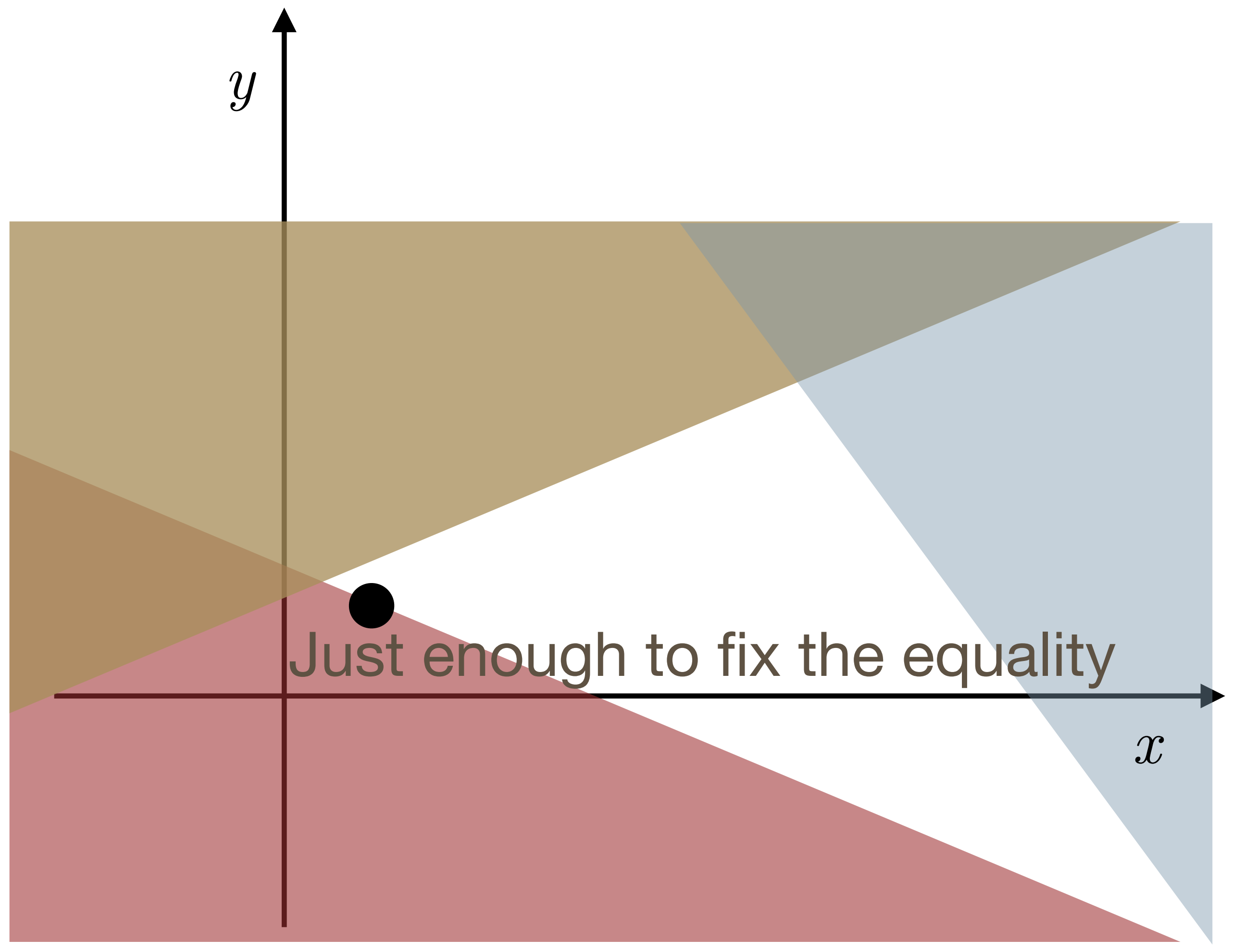


# Simplex Example



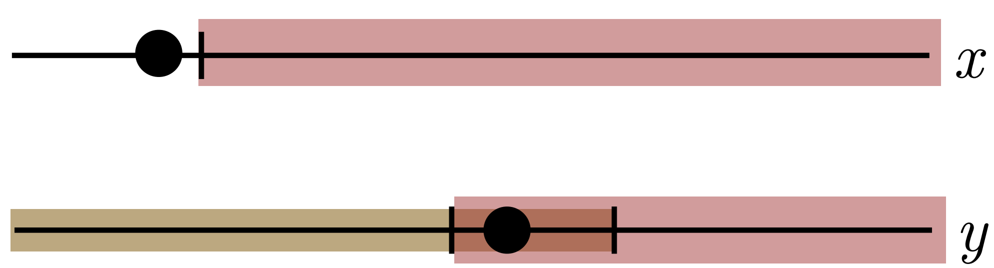
# Simplex Example

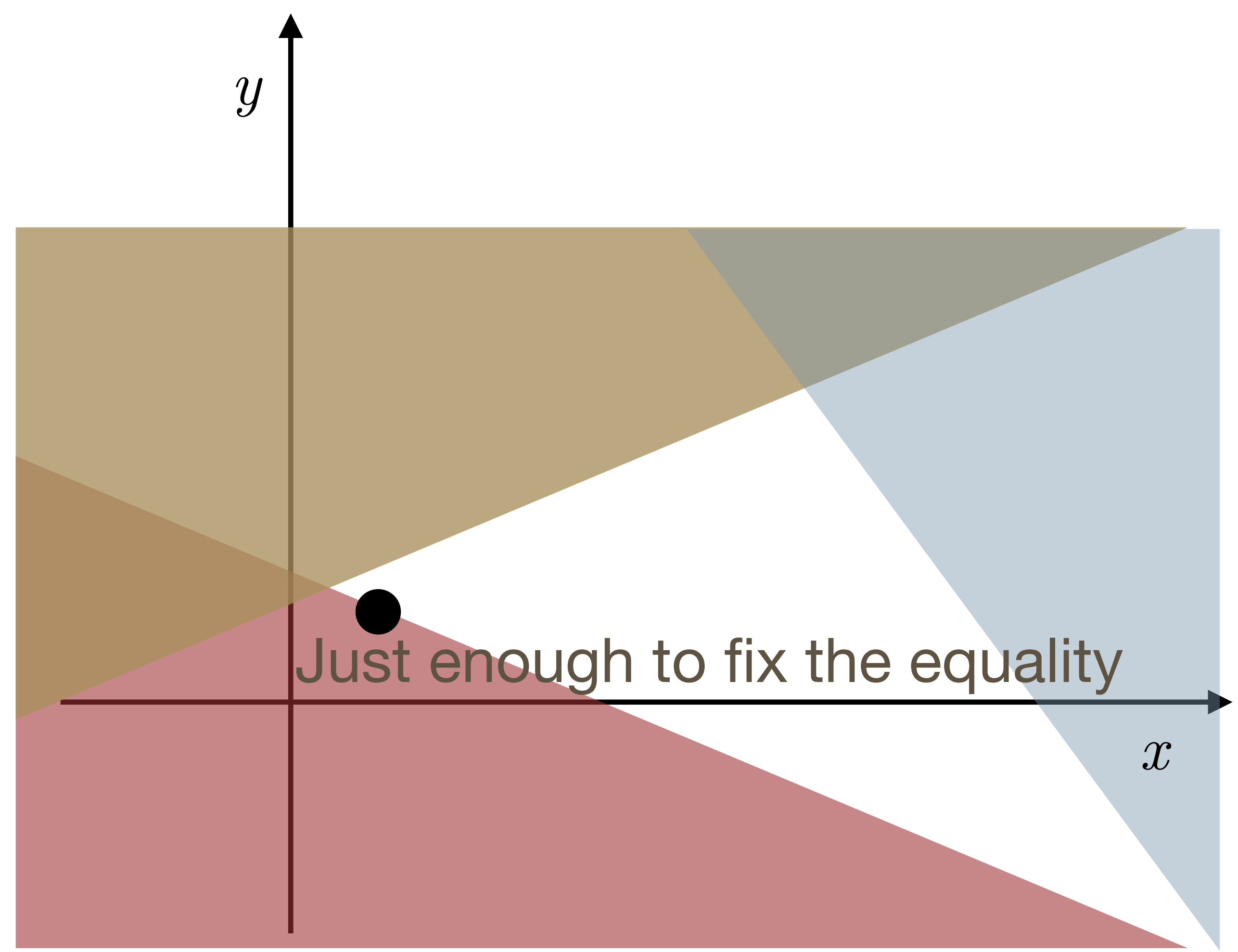



# Simplex Example

$$u = 0.5x + 3y - 2z$$

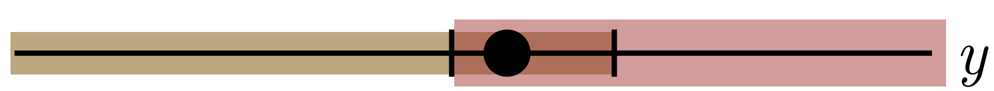



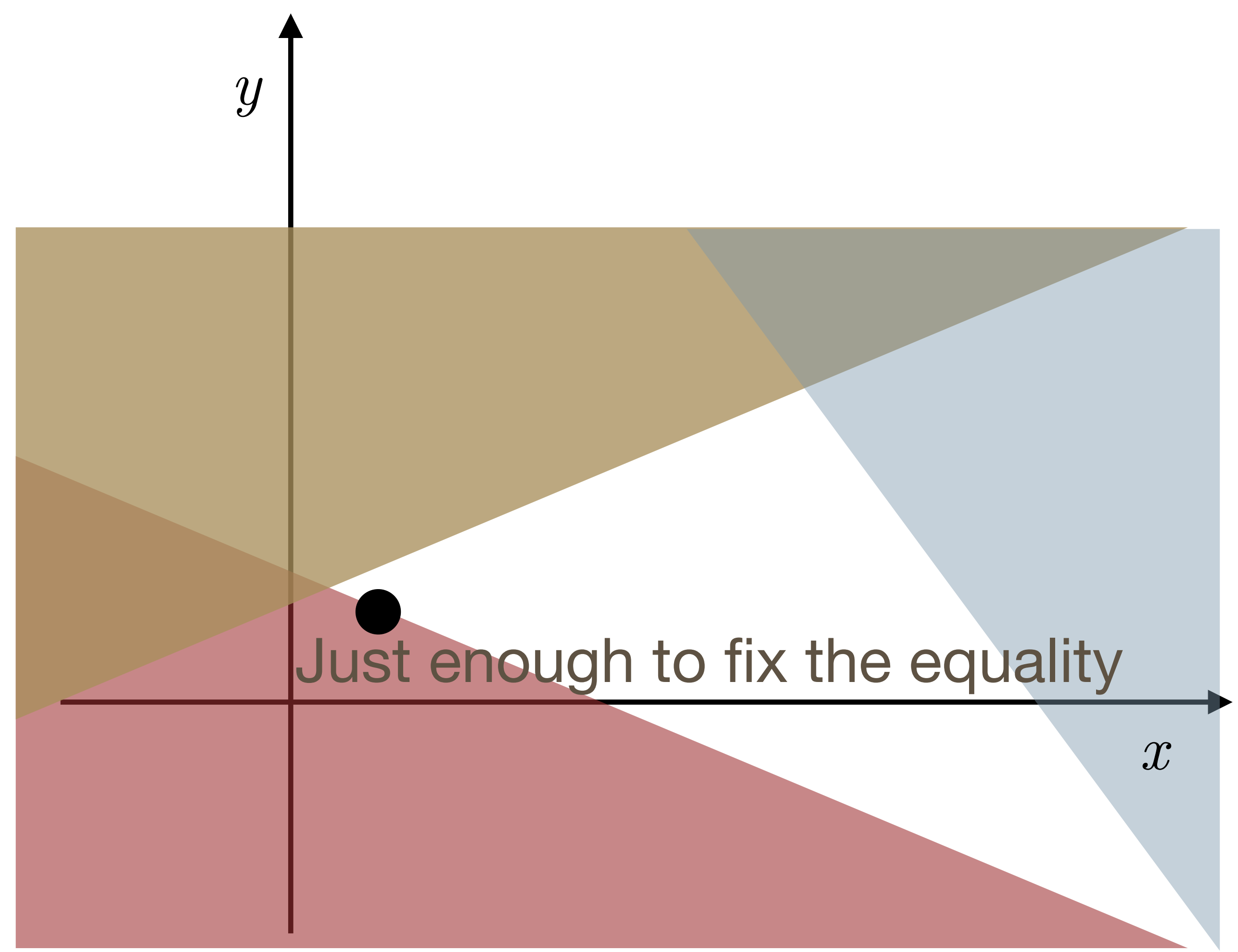



# Simplex Example

$$u = 0.5x + 3y - 2z$$

$$x = 2u - 6y + 4z$$

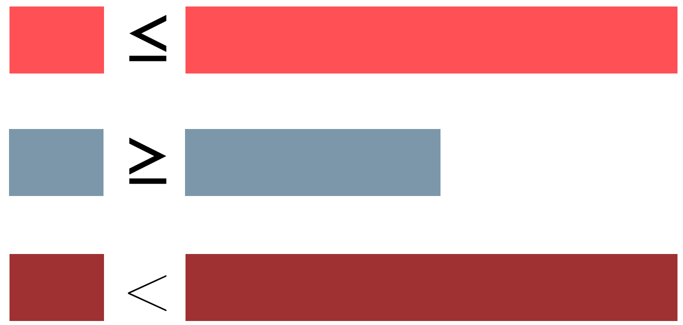



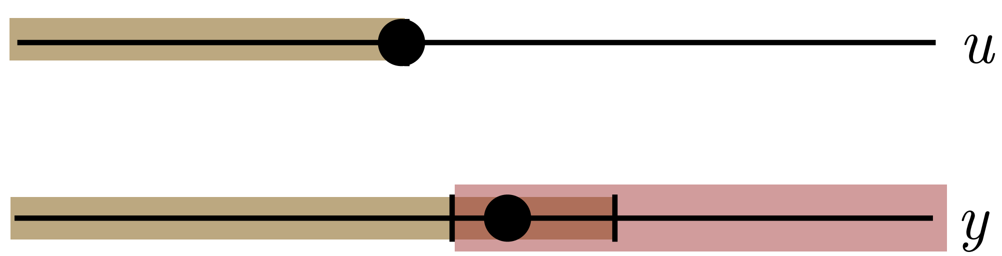



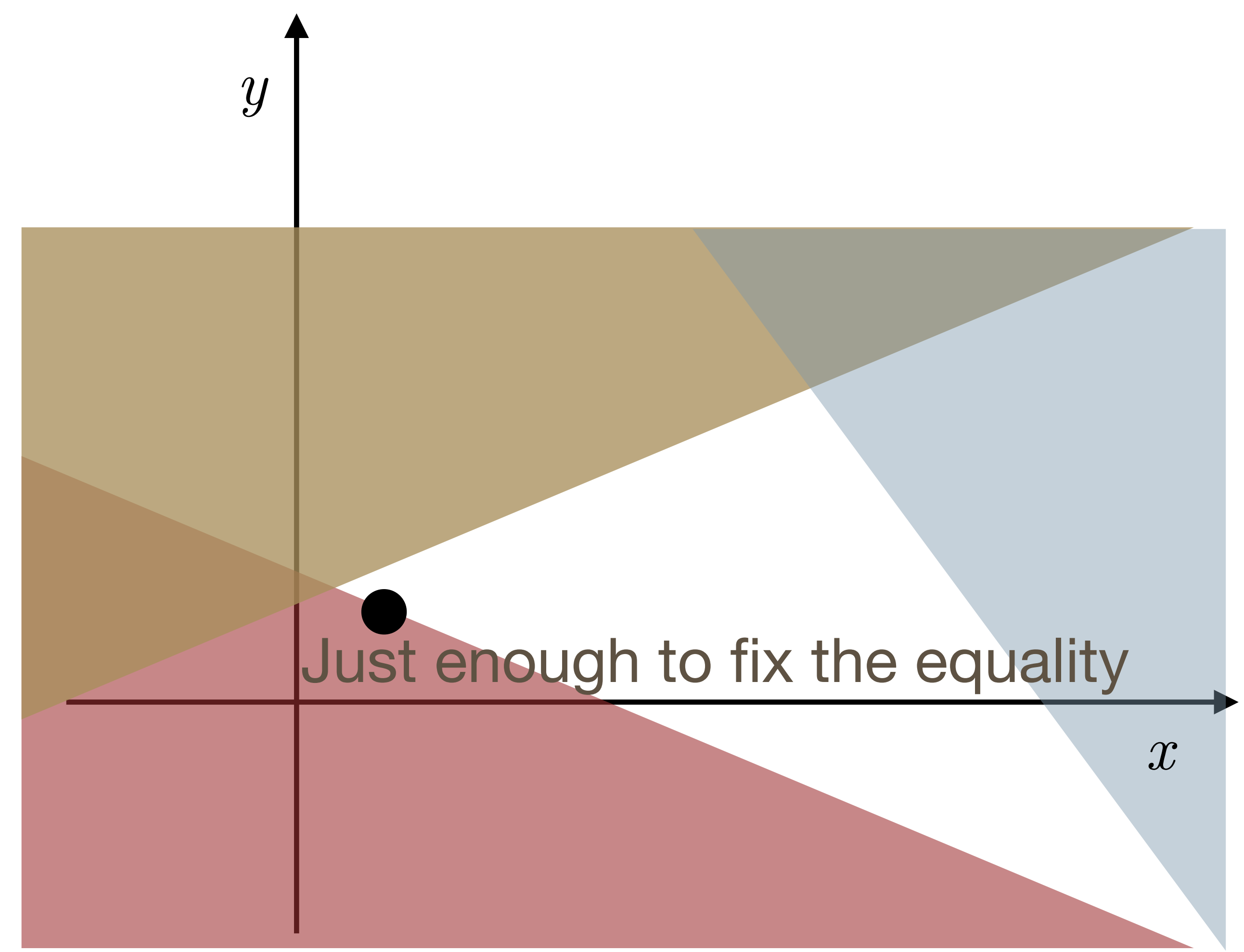
# Simplex Example

$$u = 0.5x + 3y - 2z$$

$$x = 2u - 6y + 4z$$



# Simplex Example

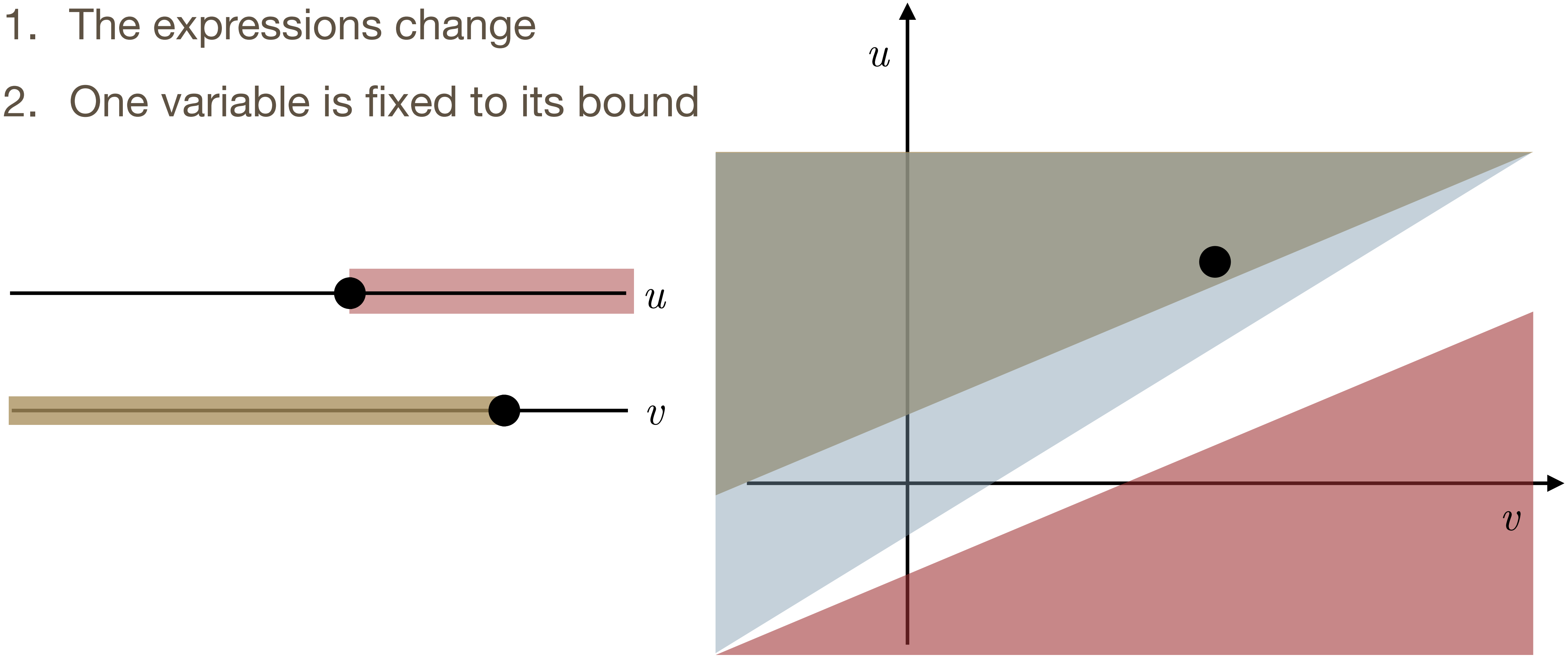
After each adjustment

1. The expressions change
2. One variable is fixed to its bound

# Simplex Example

After each adjustment

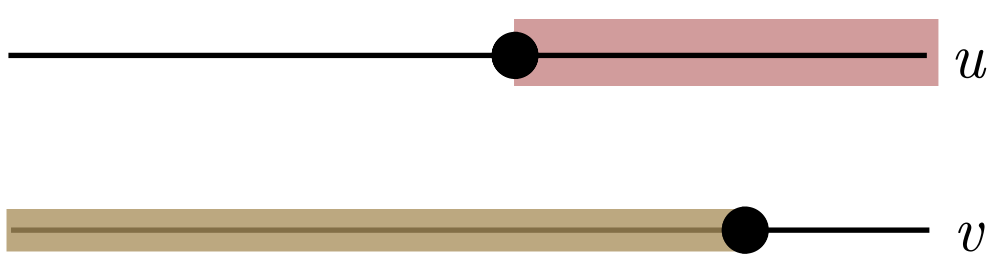
1. The expressions change
2. One variable is fixed to its bound

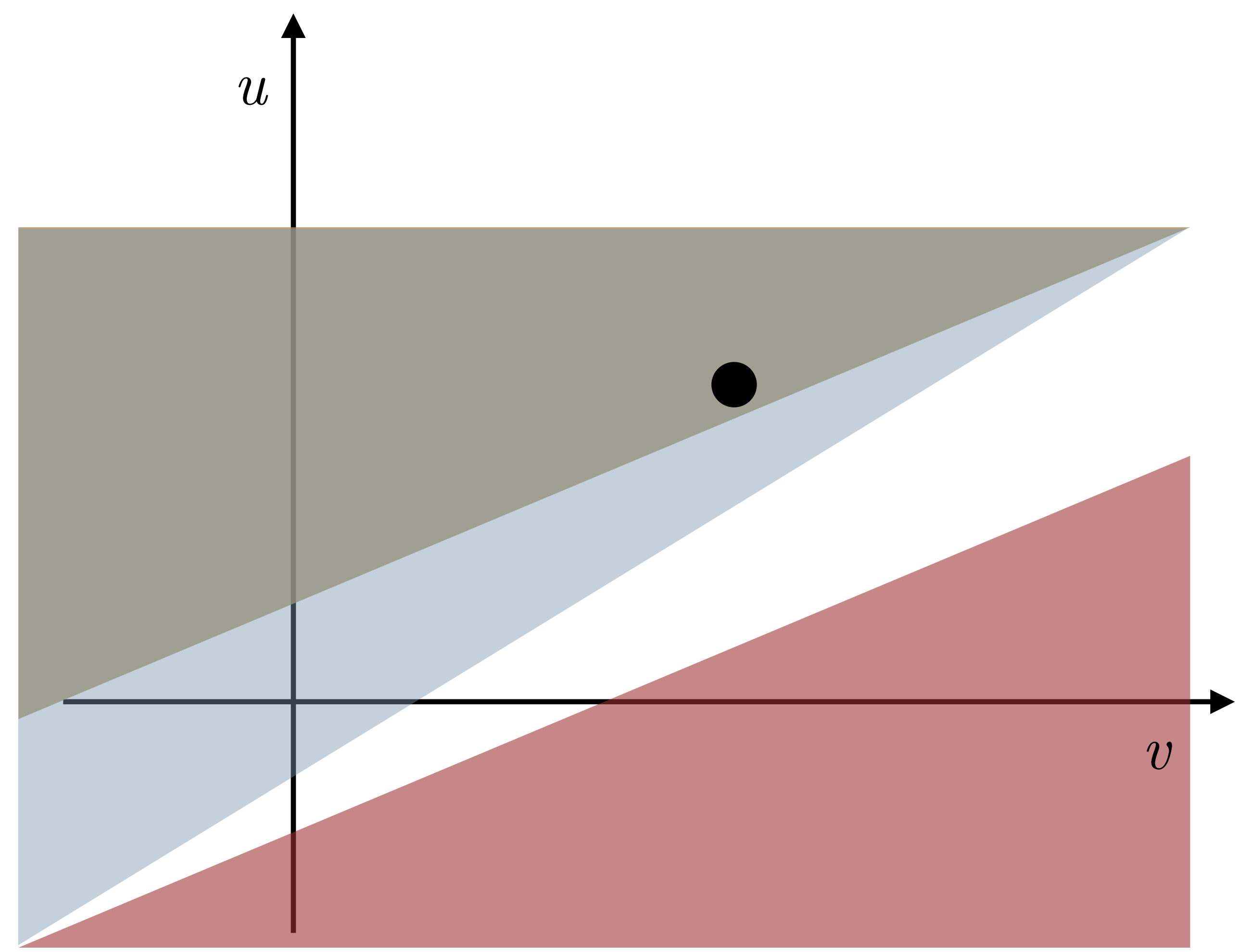


# Simplex Example

After each adjustment

1. The expressions change
2. One variable is fixed to its bound



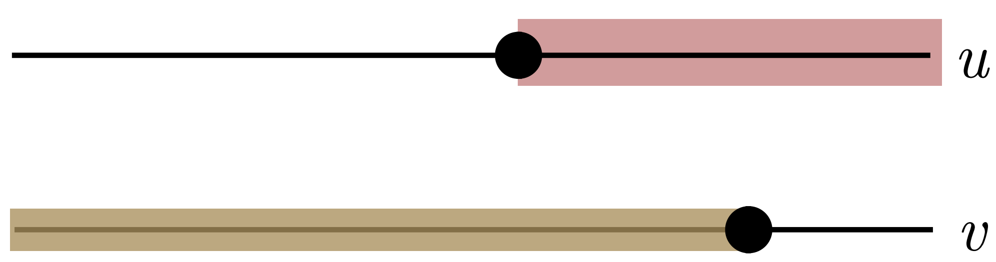

If an expression bound cannot be satisfied since the variables are at their bounds, the problem is unsatisfiable

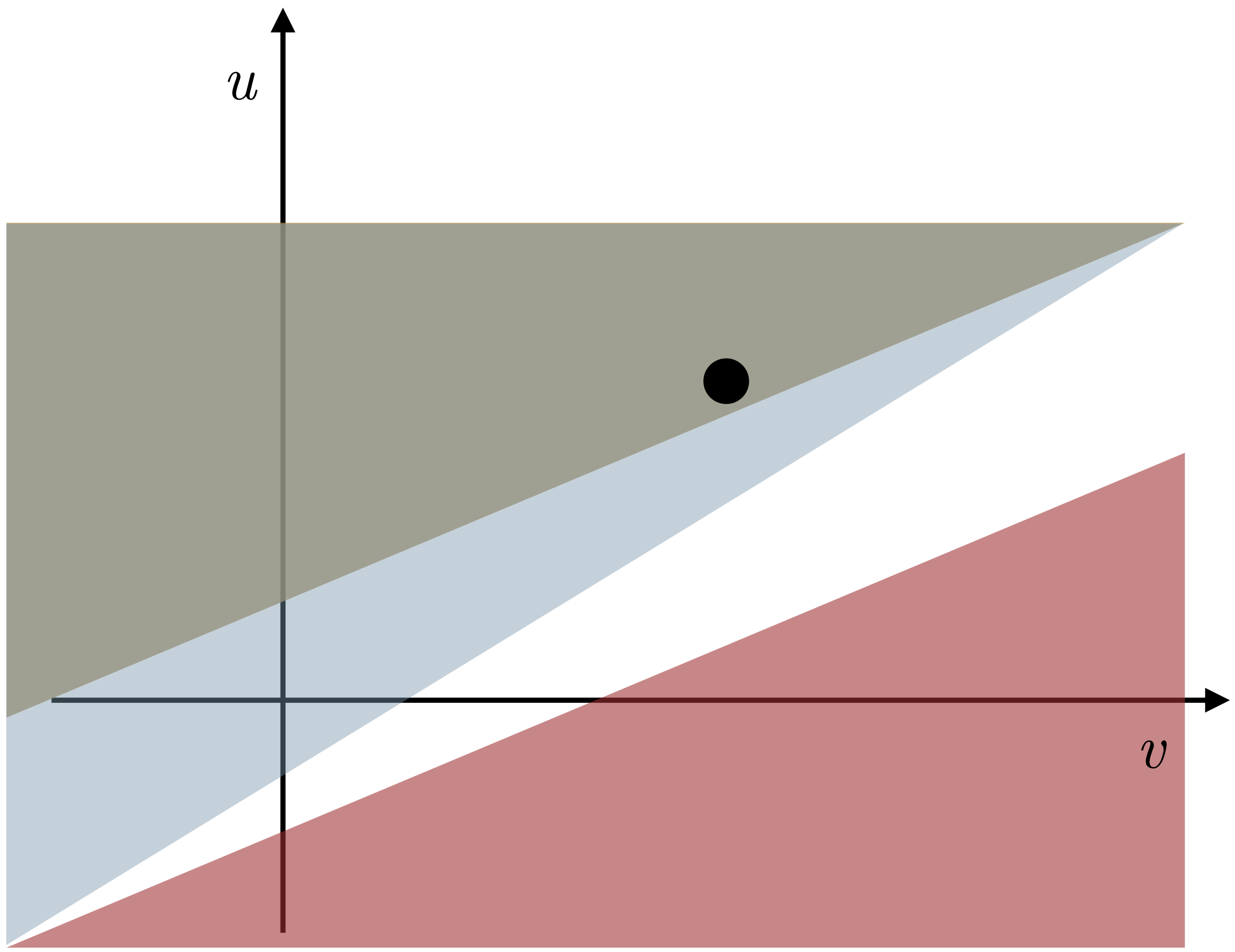
# Simplex Example

A conflict is the unsatisfied expression and the set of expressions currently bounding its variables.

After each adjustment

1. The expressions change
2. One variable is fixed to its bound



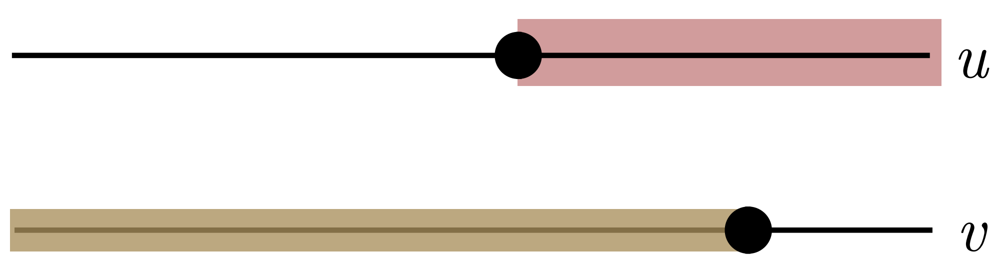

If an expression bound cannot be satisfied since the variables are at their bounds, the problem is unsatisfiable

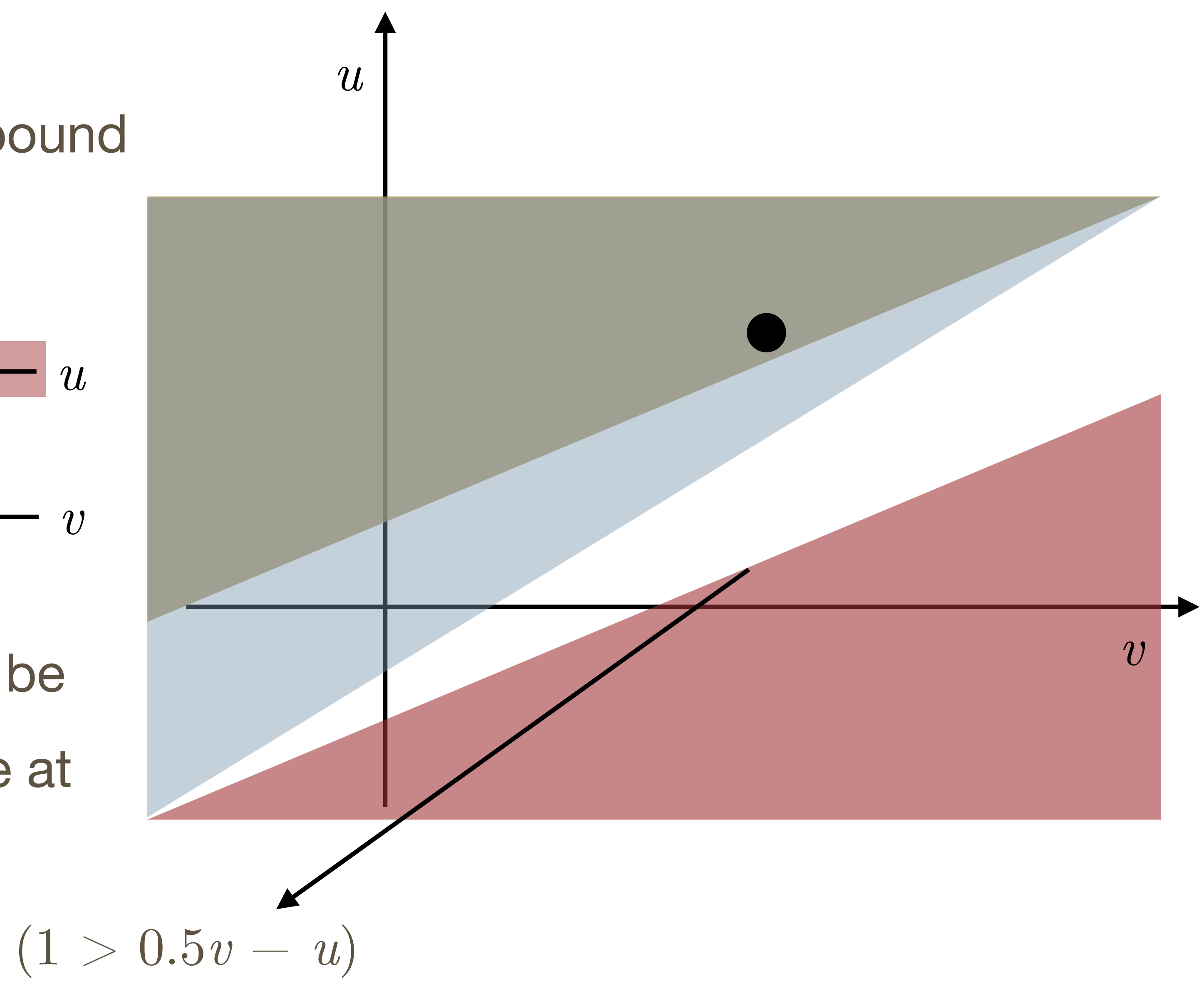
# Simplex Example

A conflict is the unsatisfied expression and the set of expressions currently bounding its variables.

After each adjustment

1. The expressions change
2. One variable is fixed to its bound



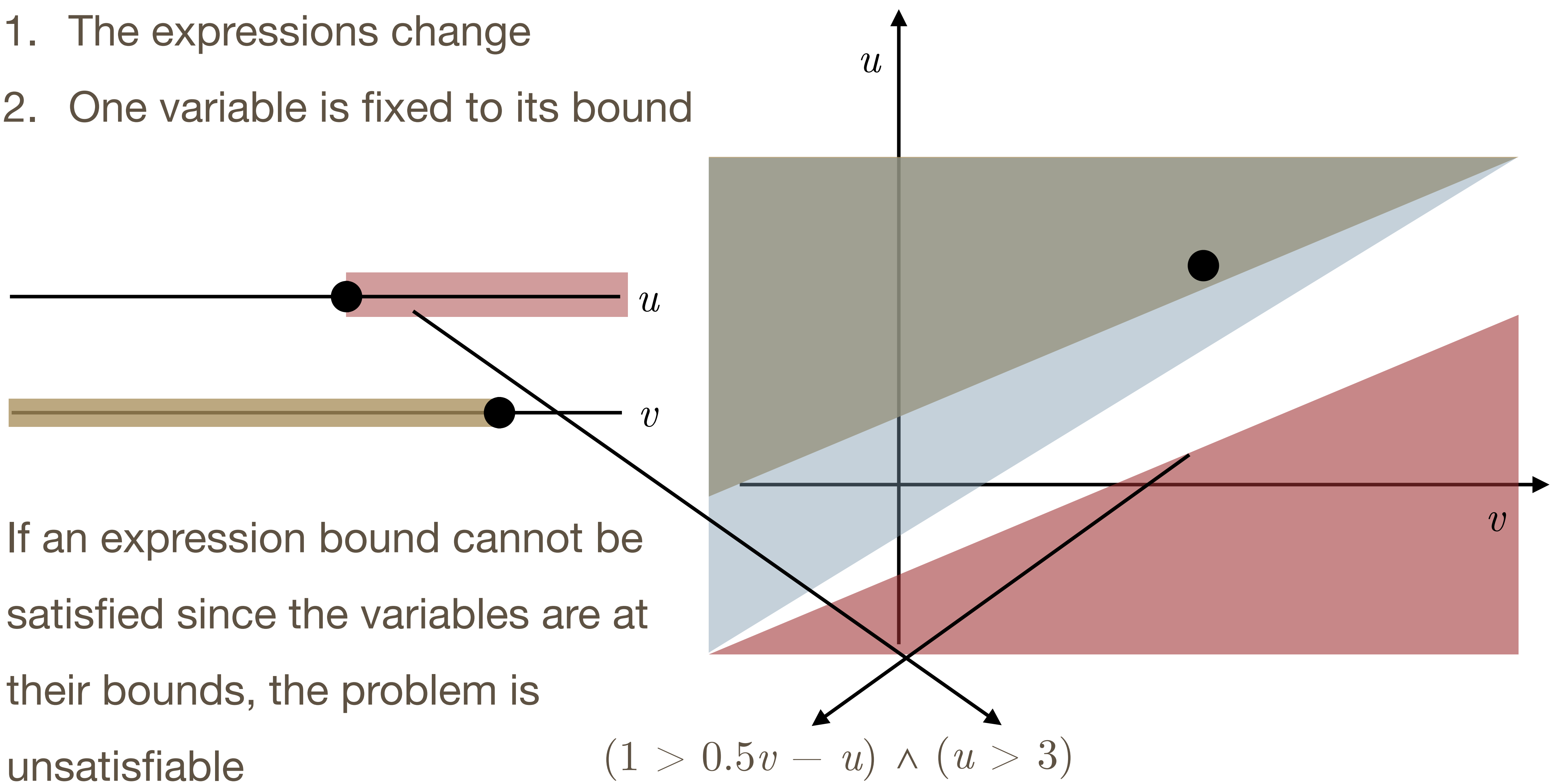

If an expression bound cannot be satisfied since the variables are at their bounds, the problem is unsatisfiable

# Simplex Example

A conflict is the unsatisfied expression and the set of expressions currently bounding its variables.

After each adjustment

1. The expressions change
2. One variable is fixed to its bound



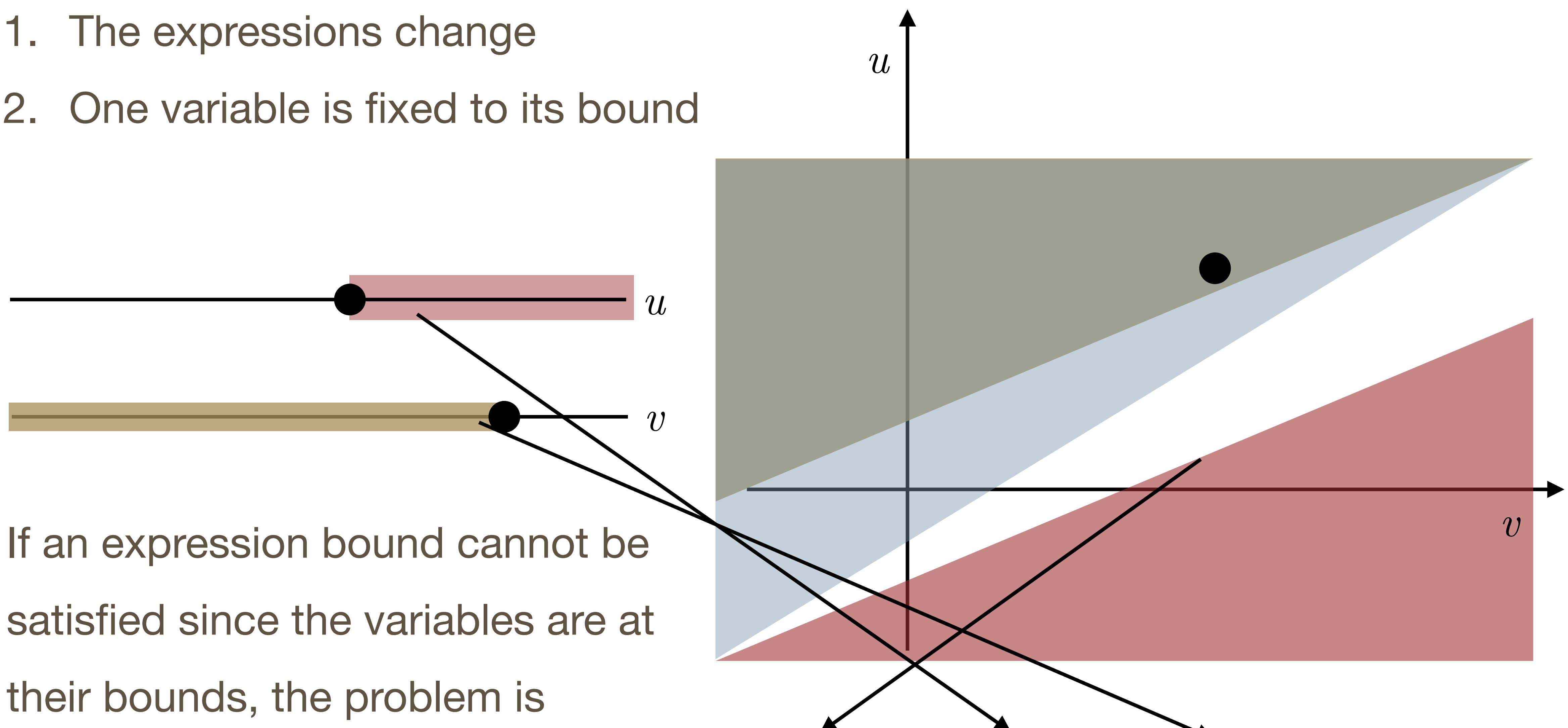

If an expression bound cannot be satisfied since the variables are at their bounds, the problem is unsatisfiable

# Simplex Example

A conflict is the unsatisfied expression and the set of expressions currently bounding its variables.

After each adjustment

1. The expressions change
2. One variable is fixed to its bound




$(1 > 0.5v - u) \wedge (u > 3) \wedge (v < 3)$

If an expression bound cannot be satisfied since the variables are at their bounds, the problem is unsatisfiable

# LRA Interpolation

Assume that the expression

bound that could not be satisfied

was $1 > 0.5v - u$

and the bounds for the variables

$u, \ v$ were

$u > 3$

$v < 3$

Assume that $(u > 3) \in B$ and

$(v < 3) \in A$, $(1 > 0.5v - u) \in A$.

# LRA Interpolation

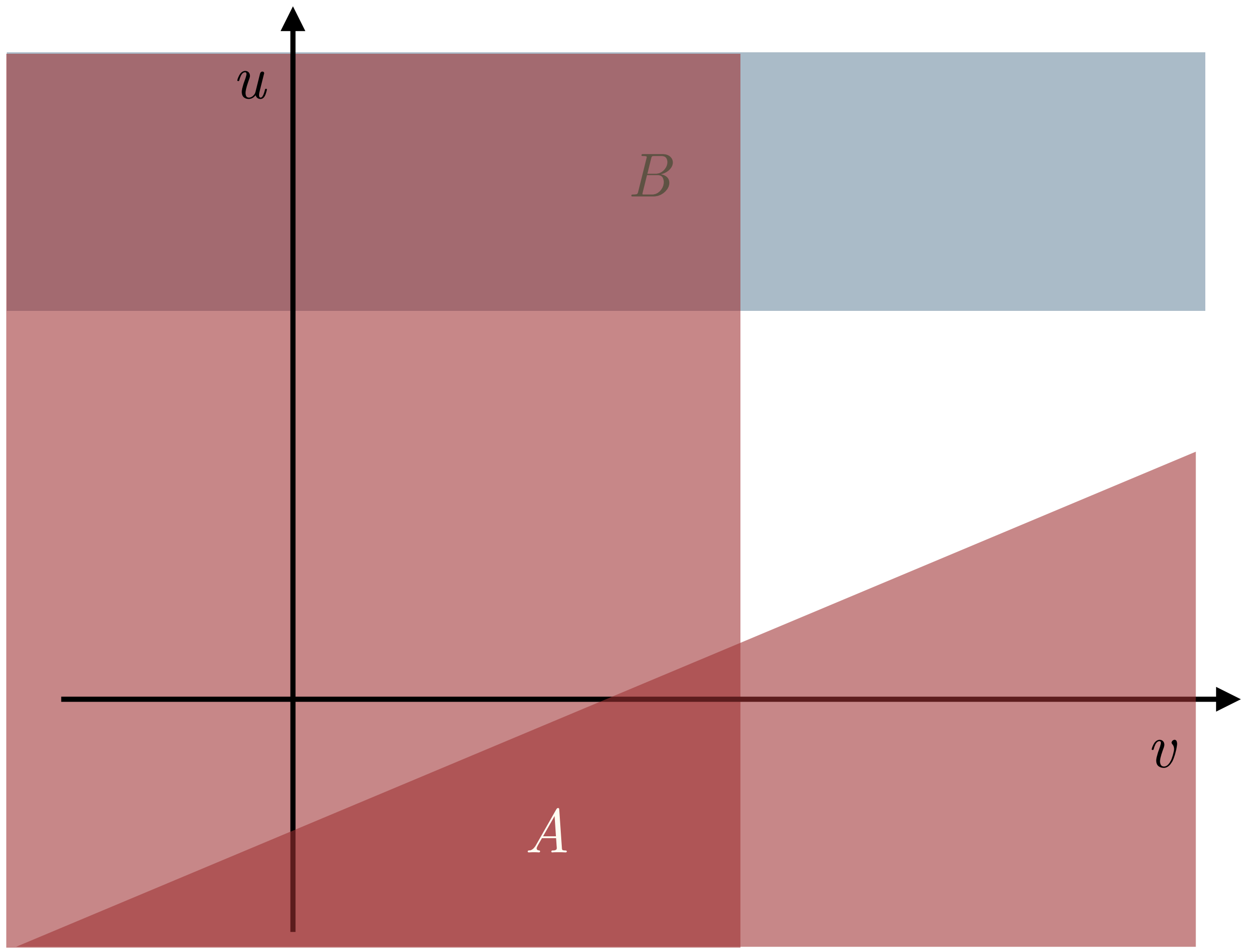
Assume that the expression bound that could not be satisfied was $1 > 0.5v - u$

and the bounds for the variables $u,\ v$ were

$u > 3$

$v < 3$

Assume that $(u > 3) \in B$ and $(v < 3) \in A$, $(1 > 0.5v - u) \in A$.



# LRA Interpolation

The interpolant for $A$ is obtained by summing to the expression the bounds in $A$ multiplied by their factors in the expression:

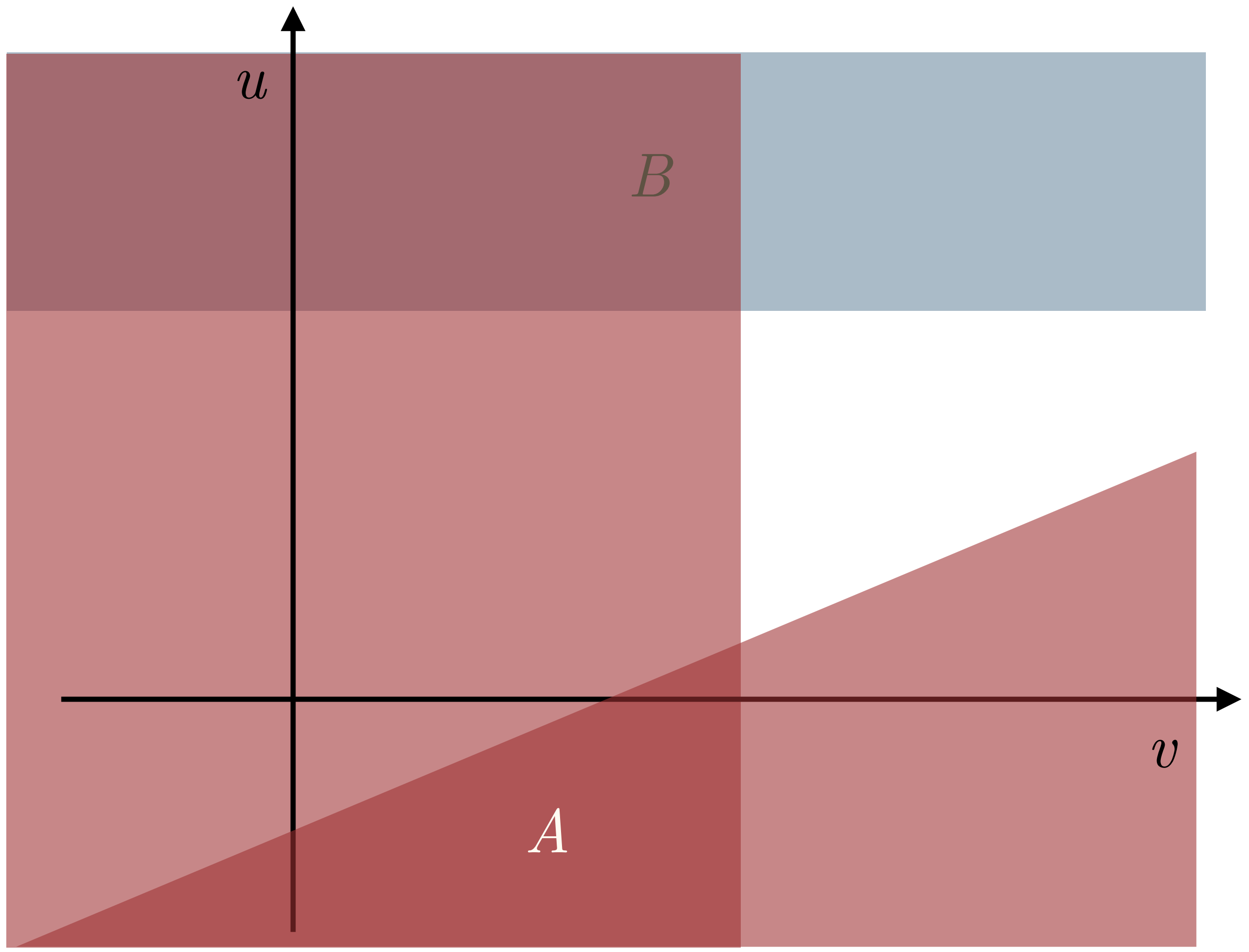
Assume that the expression bound that could not be satisfied was $1 > 0.5v - u$

and the bounds for the variables $u, \ v$ were

$u > 3$

$v < 3$

Assume that $(u > 3) \in B$ and $(v < 3) \in A, \ (1 > 0.5v - u) \in A.$



# LRA Interpolation

The interpolant for $A$ is obtained by summing to the expression the bounds in $A$ multiplied by their factors in the expression:

$(0.5v - u - 1)$

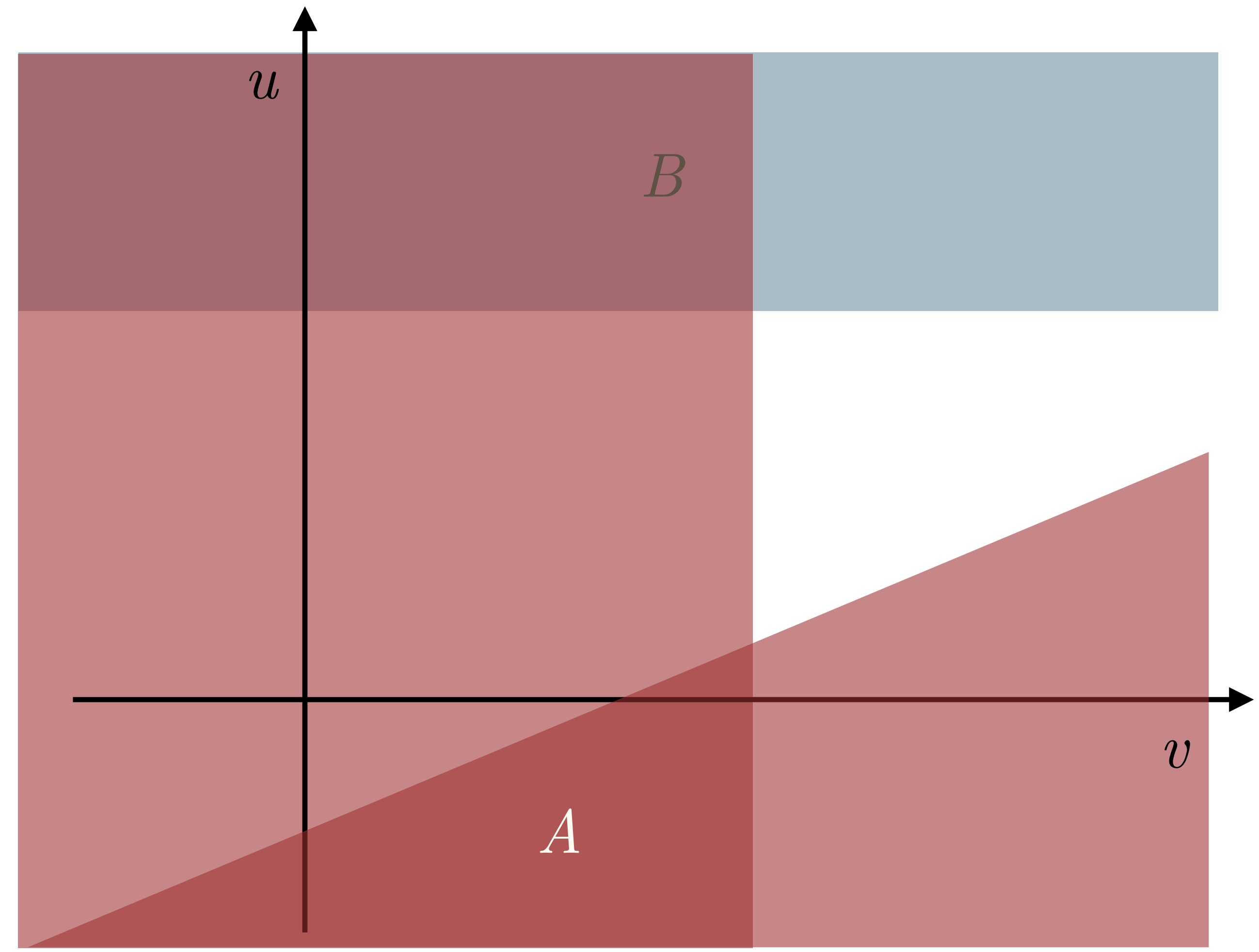
Assume that the expression bound that could not be satisfied was $1 > 0.5v - u$

and the bounds for the variables $u,\ v$ were

$u > 3$

$v < 3$

Assume that $(u > 3) \in B$ and $(v < 3) \in A$, $(1 > 0.5v - u) \in A$.



# LRA Interpolation

The interpolant for $A$ is obtained by summing to the expression the bounds in $A$ multiplied by their factors in the expression:

$$(0.5v - u - 1) + 0.5(3 - v)$$

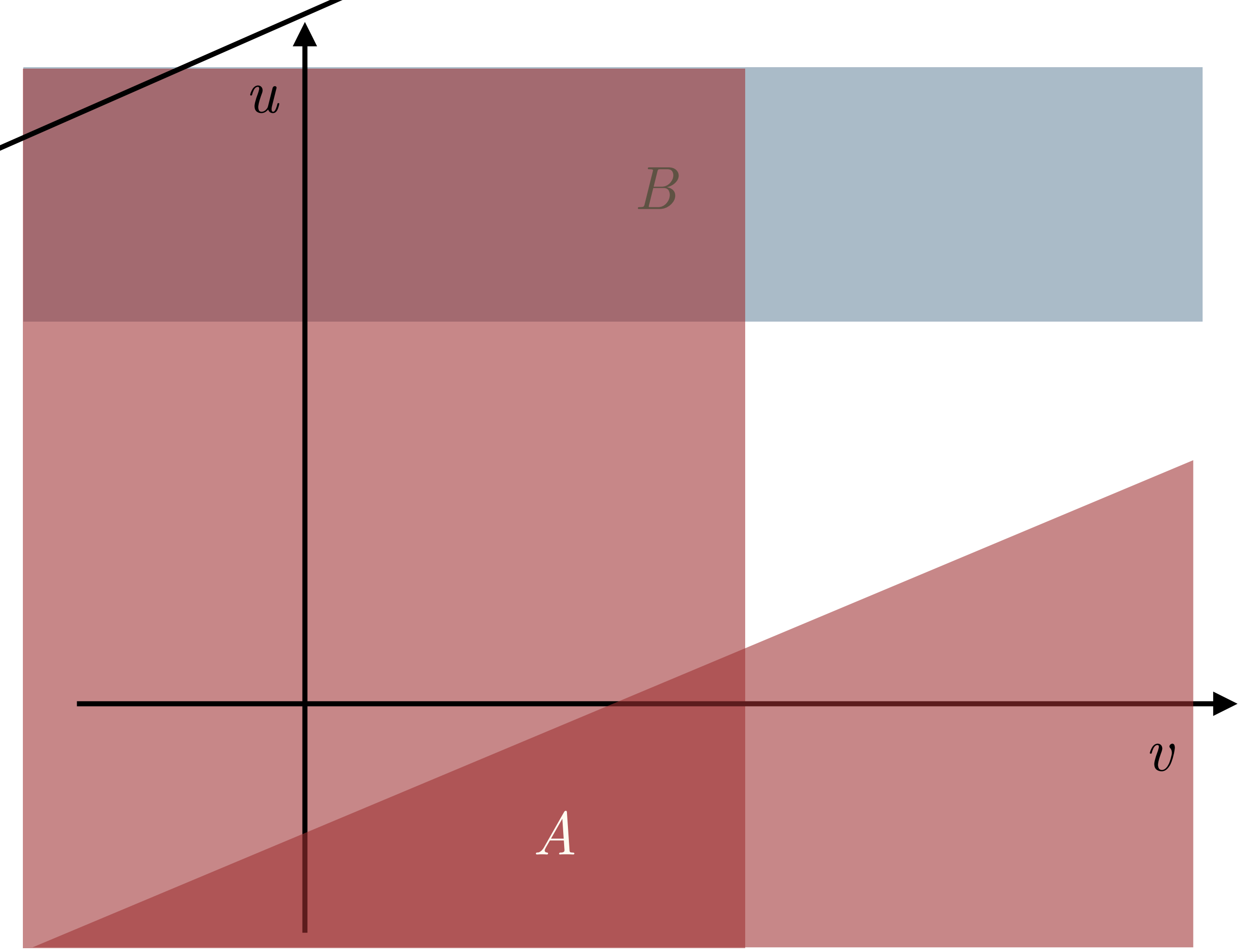
Assume that the expression bound that could not be satisfied was $1 > 0.5v - u$

and the bounds for the variables $u,\ v$ were

$u > 3$

$v < 3$

Assume that $(u > 3) \in B$ and $(v < 3) \in A$, $(1 > 0.5v - u) \in A$.



# LRA Interpolation

Assume that the expression bound that could not be satisfied was $1 > 0.5v - u$

and the bounds for the variables $u, v$ were

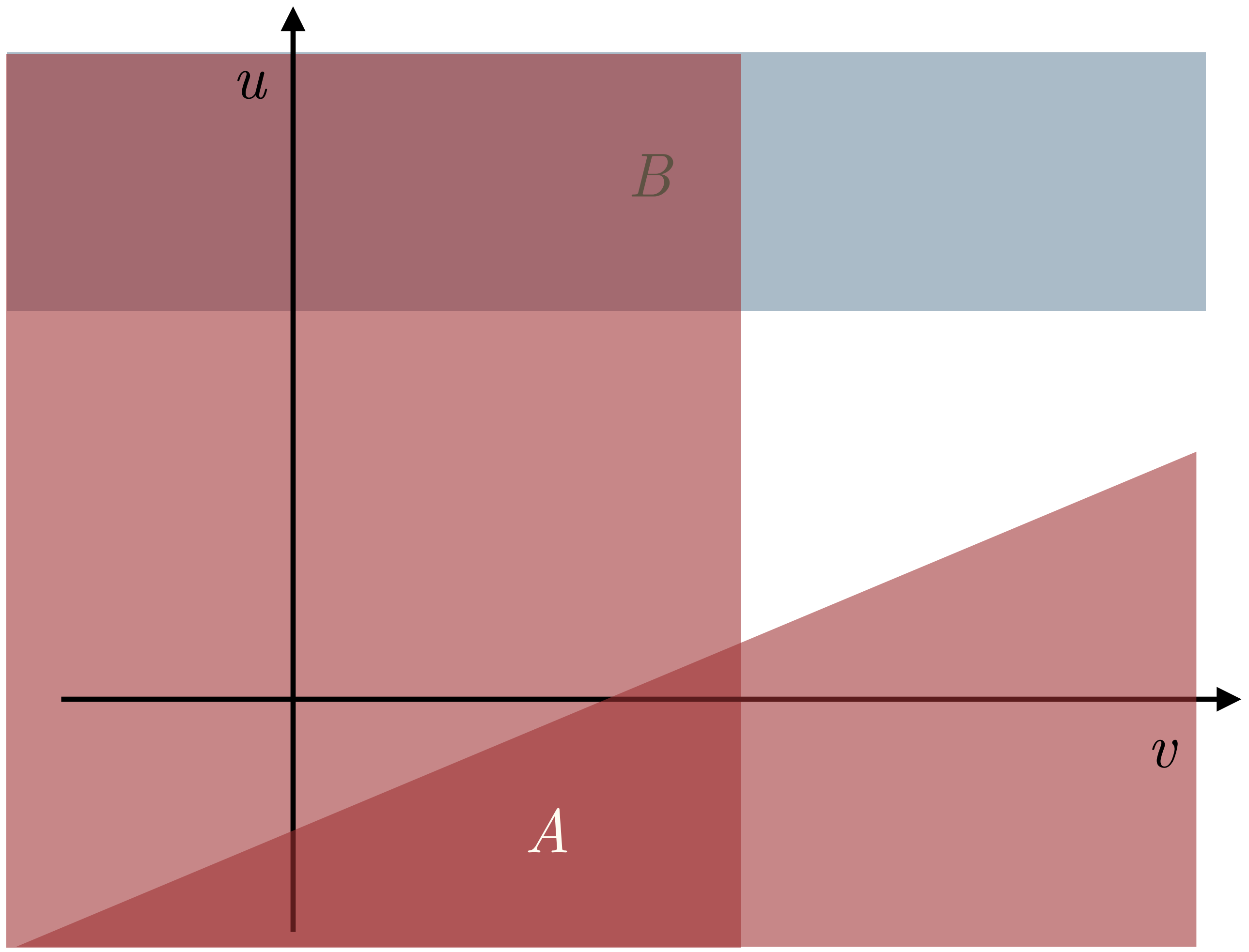
$u > 3$

$v < 3$

Assume that $(u > 3) \in B$ and $(v < 3) \in A$, $(1 > 0.5v - u) \in A$.

The interpolant for $A$ is obtained by summing to the expression the bounds in $A$ multiplied by their factors in the expression:

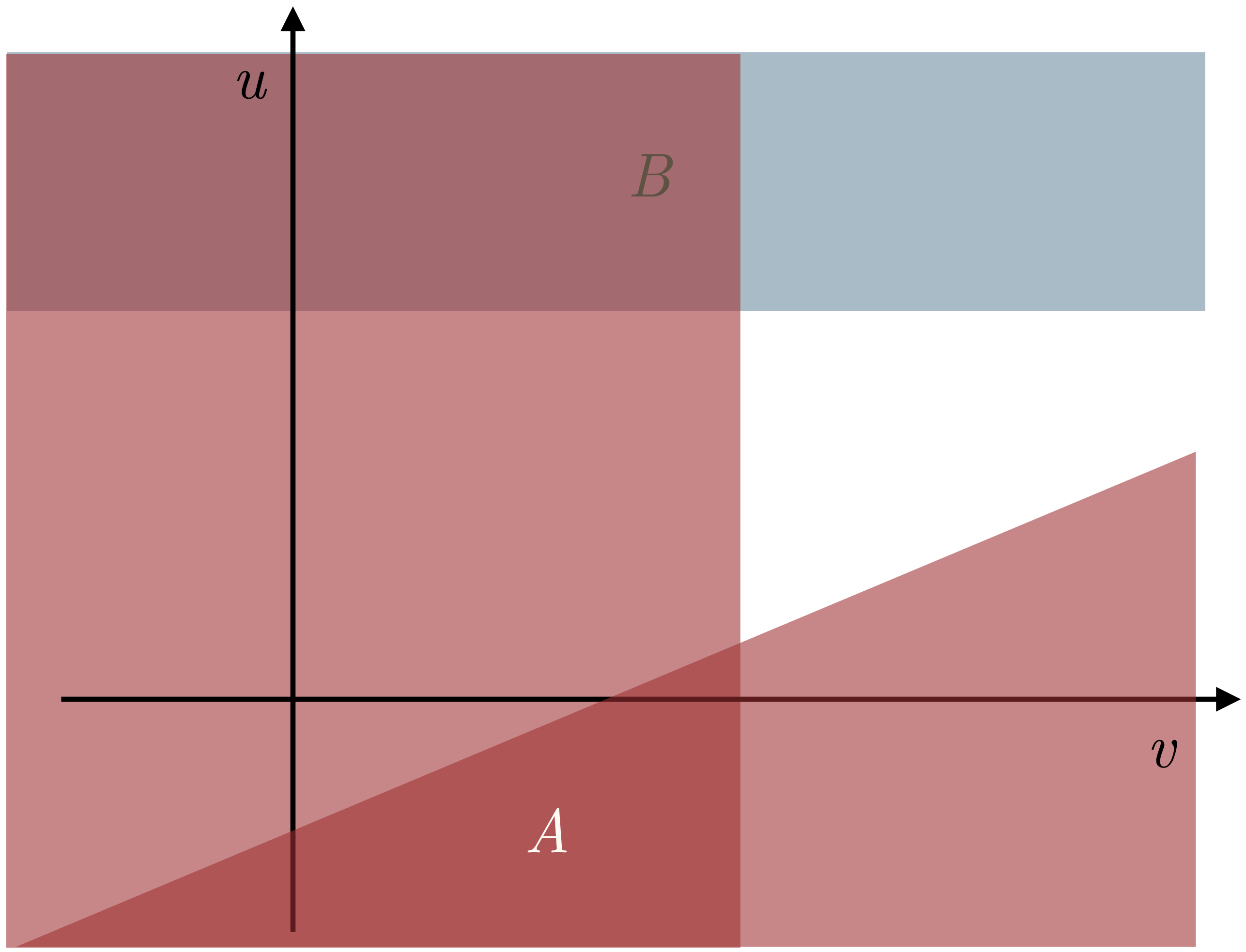
$$(0.5v - u - 1) + 0.5(3 - v)$$



# LRA Interpolation

The interpolant for $A$ is obtained by summing to the expression the bounds in $A$ multiplied by their factors in the expression:

$-u - 1 + 1.5$




Assume that the expression bound that could not be satisfied was $1 > 0.5v - u$

and the bounds for the variables $u,\ v$ were

$u > 3$

$v < 3$

Assume that $(u > 3) \in B$ and $(v < 3) \in A$, $(1 > 0.5v - u) \in A$.

# LRA Interpolation

Assume that the expression bound that could not be satisfied was $1 > 0.5v - u$ and the bounds for the variables $u, \ v$ were

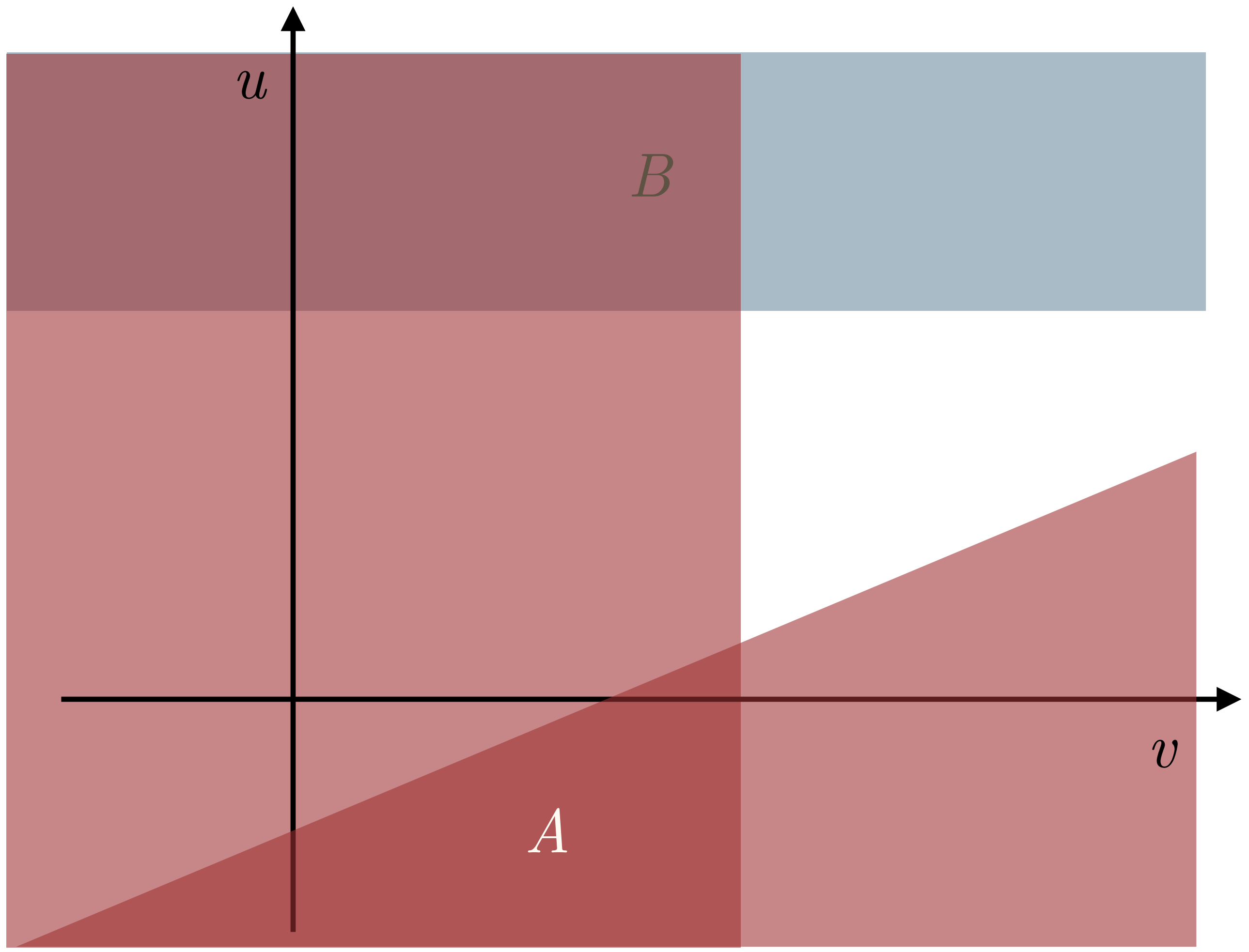
$u > 3$

$v < 3$

Assume that $(u > 3) \in B$ and $(v < 3) \in A, (1 > 0.5v - u) \in A.$

The interpolant for $A$ is obtained by summing to the expression the bounds in $A$ multiplied by their factors in the expression:

$0 <$ -$u - 1 + 1.5$



# LRA Interpolation

Assume that the expression bound that could not be satisfied was $1 > 0.5v - u$

and the bounds for the variables $u,\ v$ were

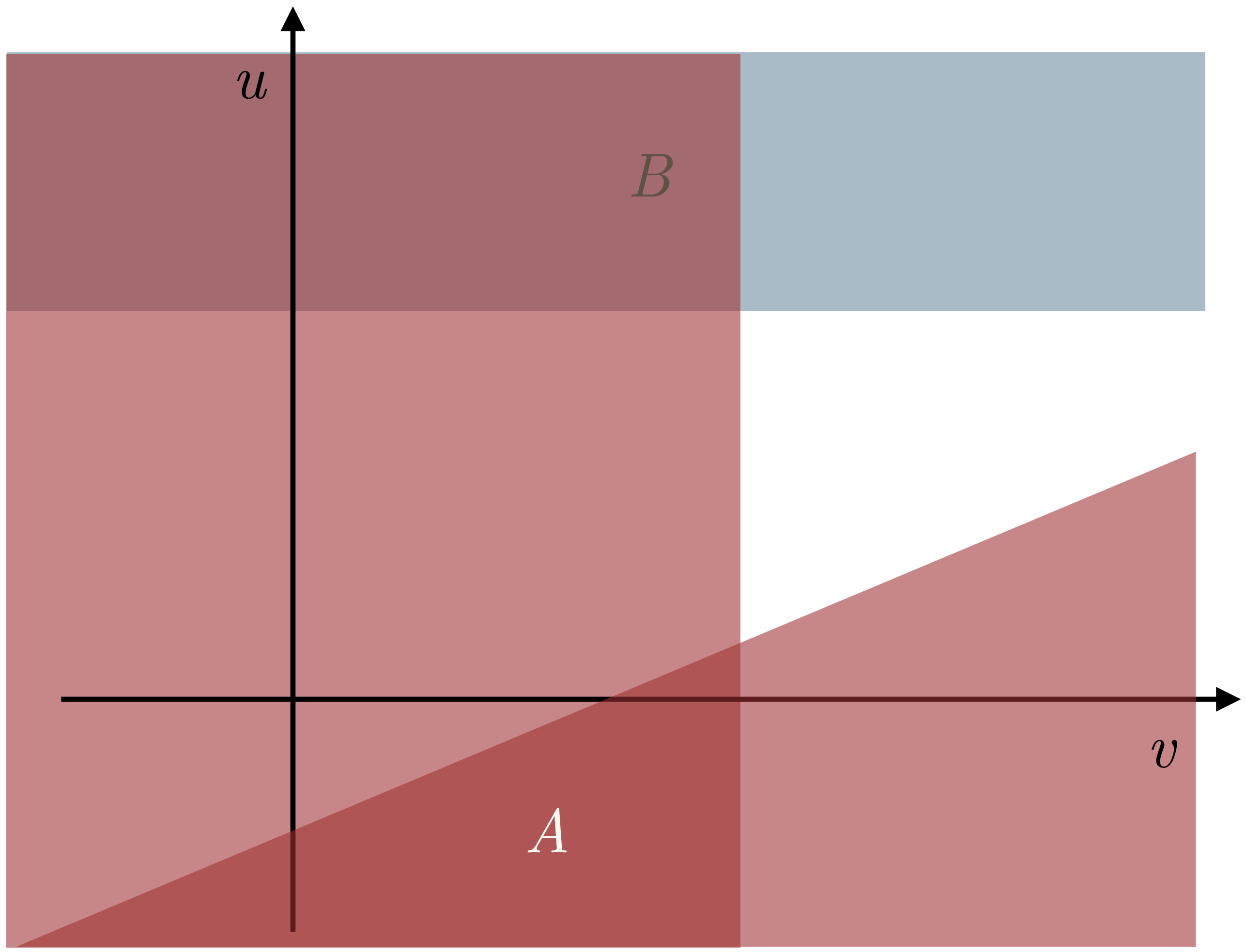
$u > 3$

$v < 3$

Assume that $(u > 3) \in B$ and $(v < 3) \in A,\ (1 > 0.5v - u) \in A.$

The interpolant for $A$ is obtained by summing to the expression the bounds in $A$ multiplied by their factors in the expression:

$u < 0.5$



# LRA Interpolation

Assume that the expression bound that could not be satisfied was $1 > 0.5v - u$

and the bounds for the variables $u, v$ were

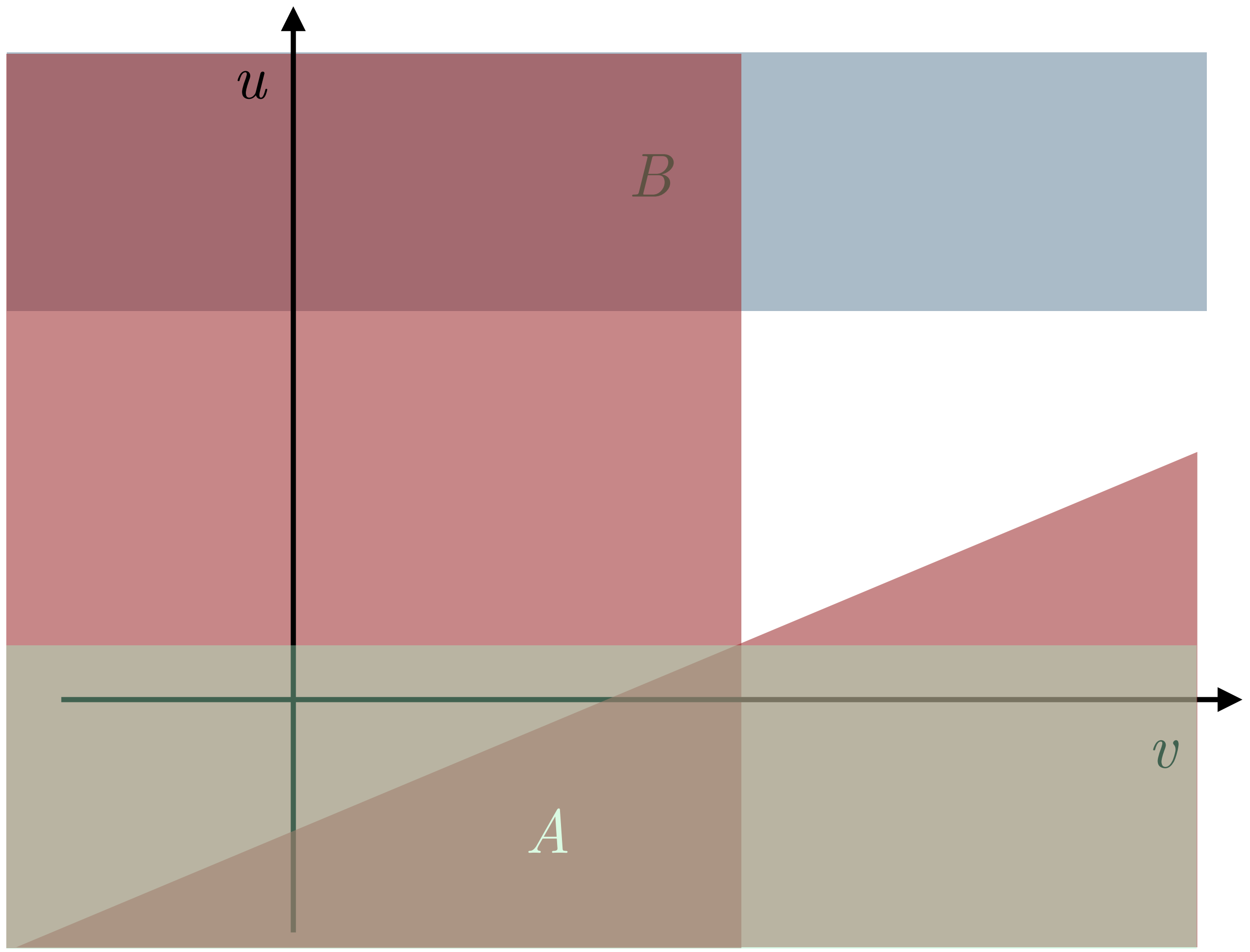
$u > 3$

$v < 3$

Assume that $(u > 3) \in B$ and $(v < 3) \in A$, $(1 > 0.5v - u) \in A$.

The interpolant for $A$ is obtained by summing to the expression the bounds in $A$ multiplied by their factors in the expression:

$u < 0.5$



# Duality-based Interpolation for LRA

Given a primal interpolant

$I = c_1 \leq t(\boldsymbol{x})$,

the dual interpolant has the form

$I' = c_2 < t(\boldsymbol{x})$

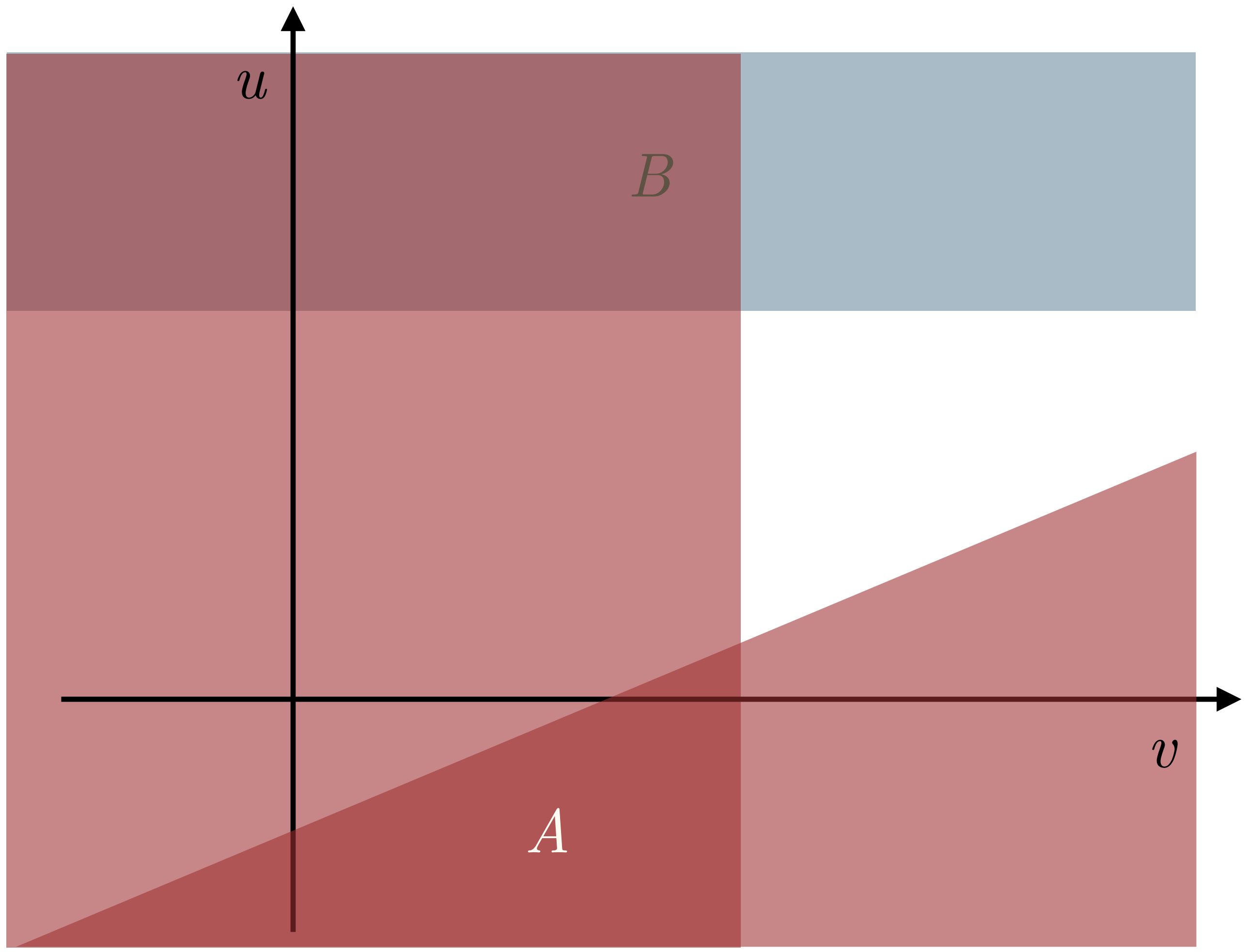
# Duality-based Interpolation for LRA

Given a primal interpolant

$I = \; c_1 \leq t(\boldsymbol{x})$,

the dual interpolant has the form

$I' = \; c_2 < \; t(\boldsymbol{x})$



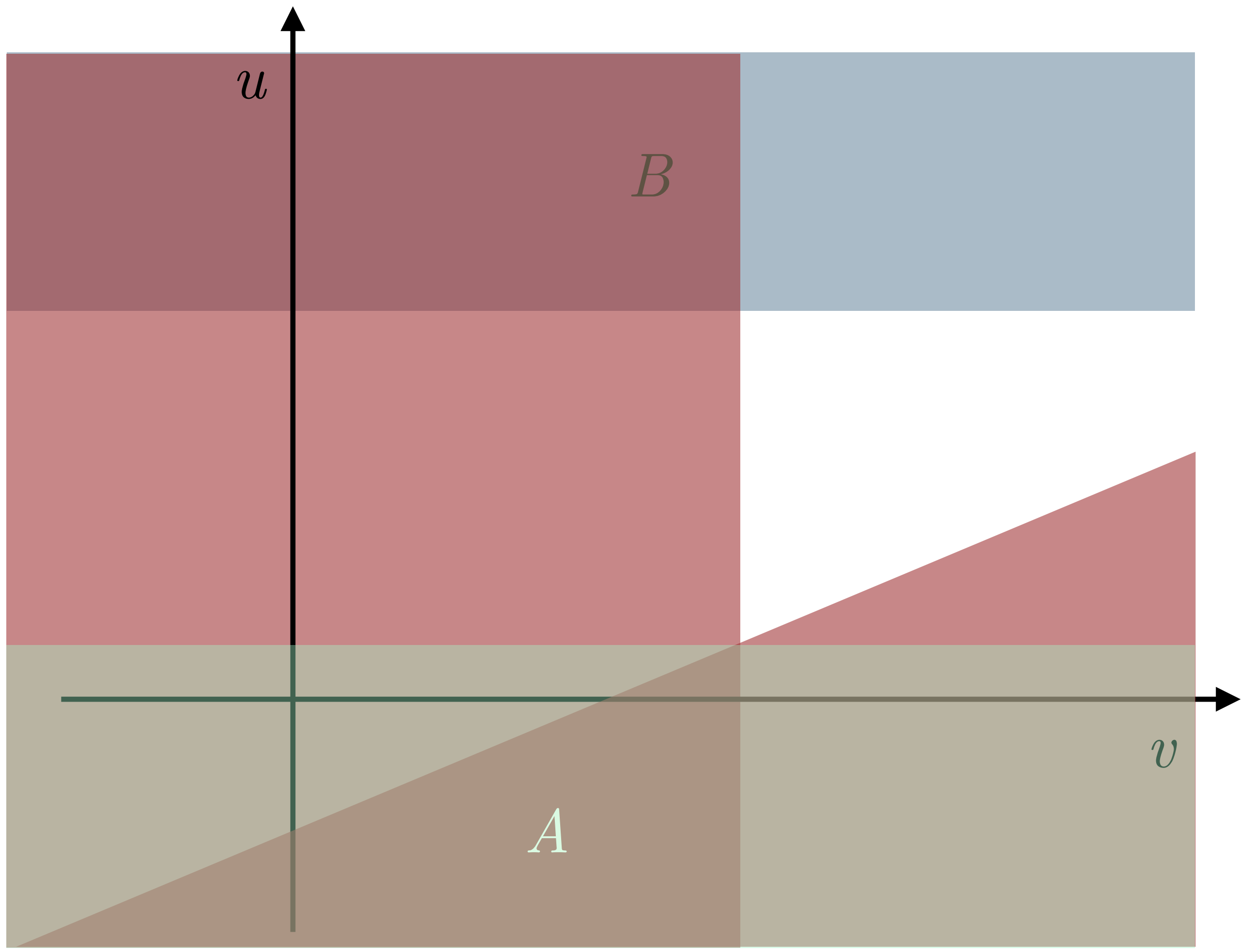
# Duality-based Interpolation for LRA

Given a primal interpolant

$I = c_1 \leq t(\boldsymbol{x})$,

the dual interpolant has the form

$I' = c_2 < t(\boldsymbol{x})$



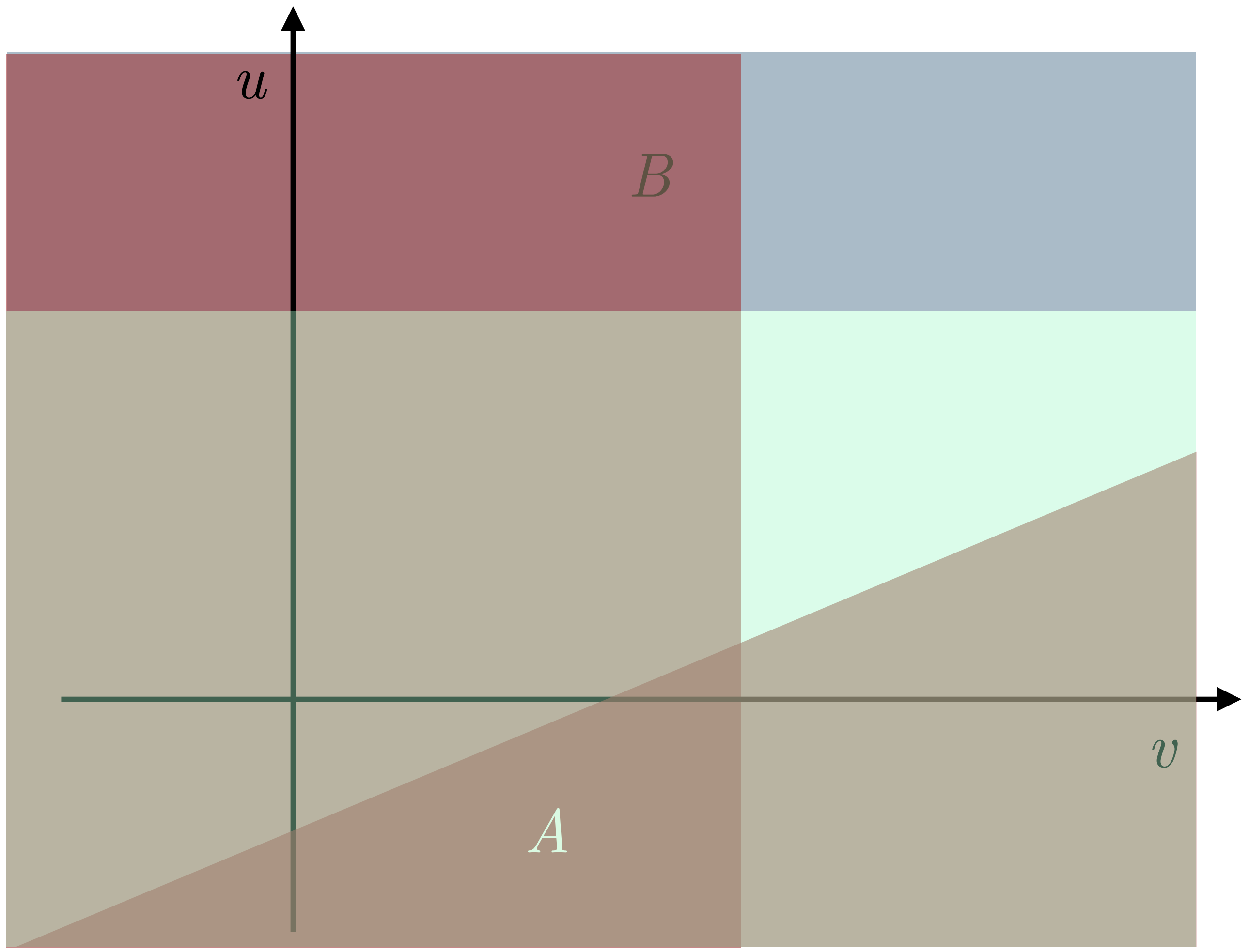
# Duality-based Interpolation for LRA

Given a primal interpolant

$I = c_1 \leq t(\boldsymbol{x})$,

the dual interpolant has the form

$I' = c_2 < t(\boldsymbol{x})$



# Duality-based Interpolation for LRA

Given a primal interpolant

$I = c_1 \leq t(\boldsymbol{x}),$

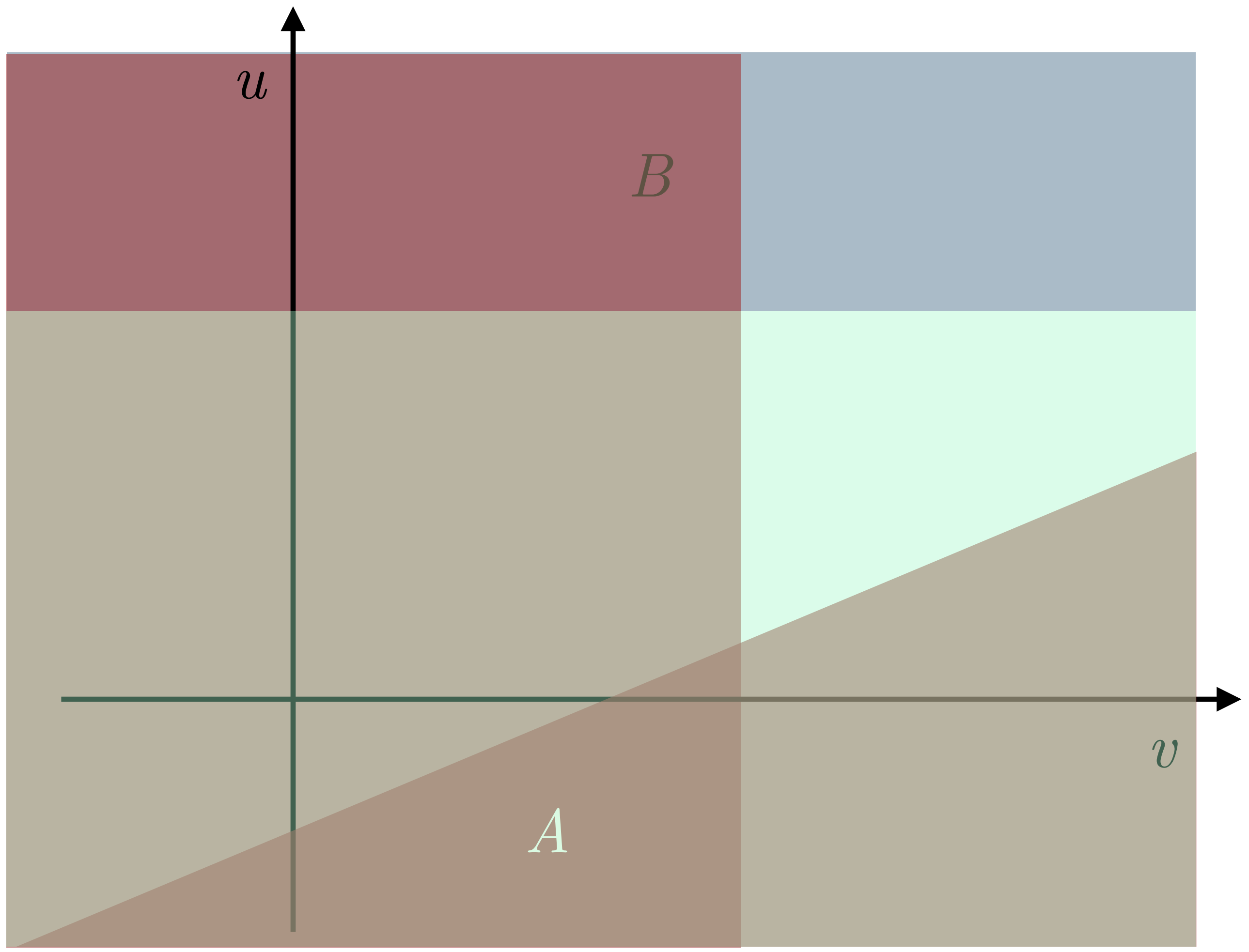
the dual interpolant has the form

$I' = c_2 < t(\boldsymbol{x})$

All the inequalities of the form

$c < t(\boldsymbol{x}),\ c_1 \leq c < c_2$

are also interpolants for $A$



# Duality-based Interpolation for LRA

Given a primal interpolant

$I = c_1 \leq t(\boldsymbol{x}),$

the dual interpolant has the form

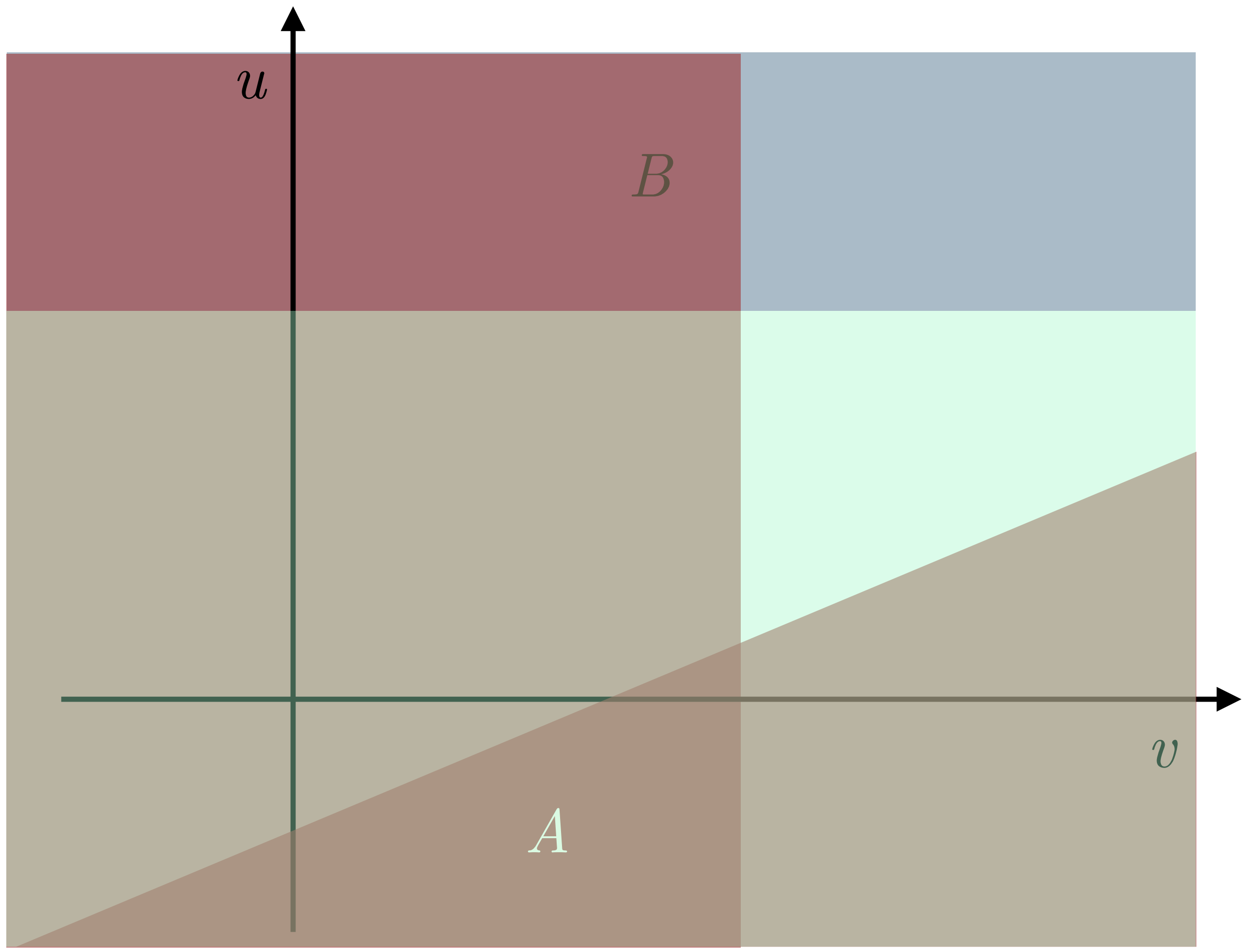
$I' = c_2 < t(\boldsymbol{x})$

All the inequalities of the form

$c < t(\boldsymbol{x}),\ c_1 \leq c < c_2$

are also interpolants for $A$

$I \longrightarrow I'$



# Duality-based Interpolation for LRA

Given a primal interpolant

$I = c_1 \le t(\boldsymbol{x})$,

the dual interpolant has the form

$I' = c_2 < t(\boldsymbol{x})$

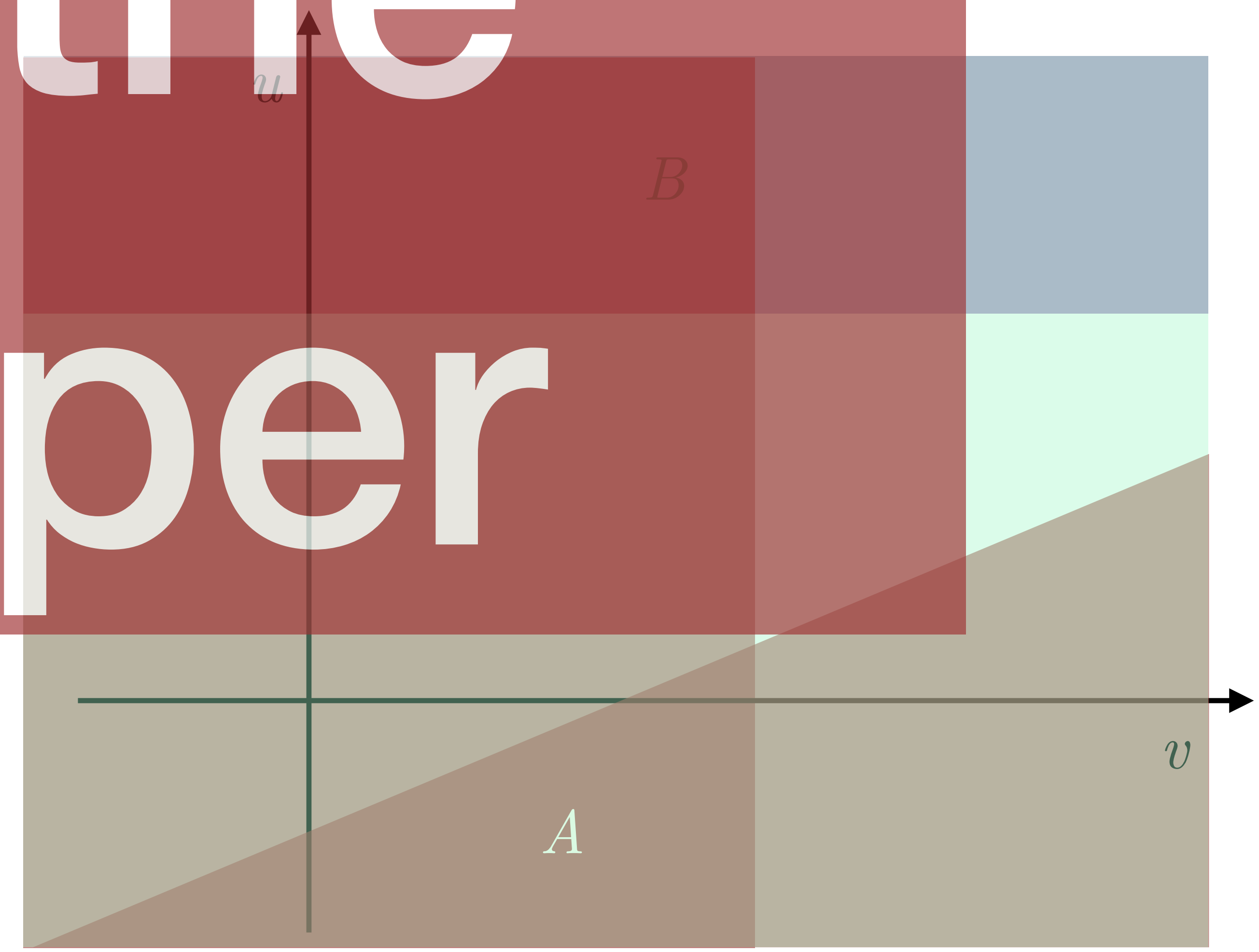
All the inequalities of the form

$c < t(\boldsymbol{x})$, $c_1 \le c < c_2$

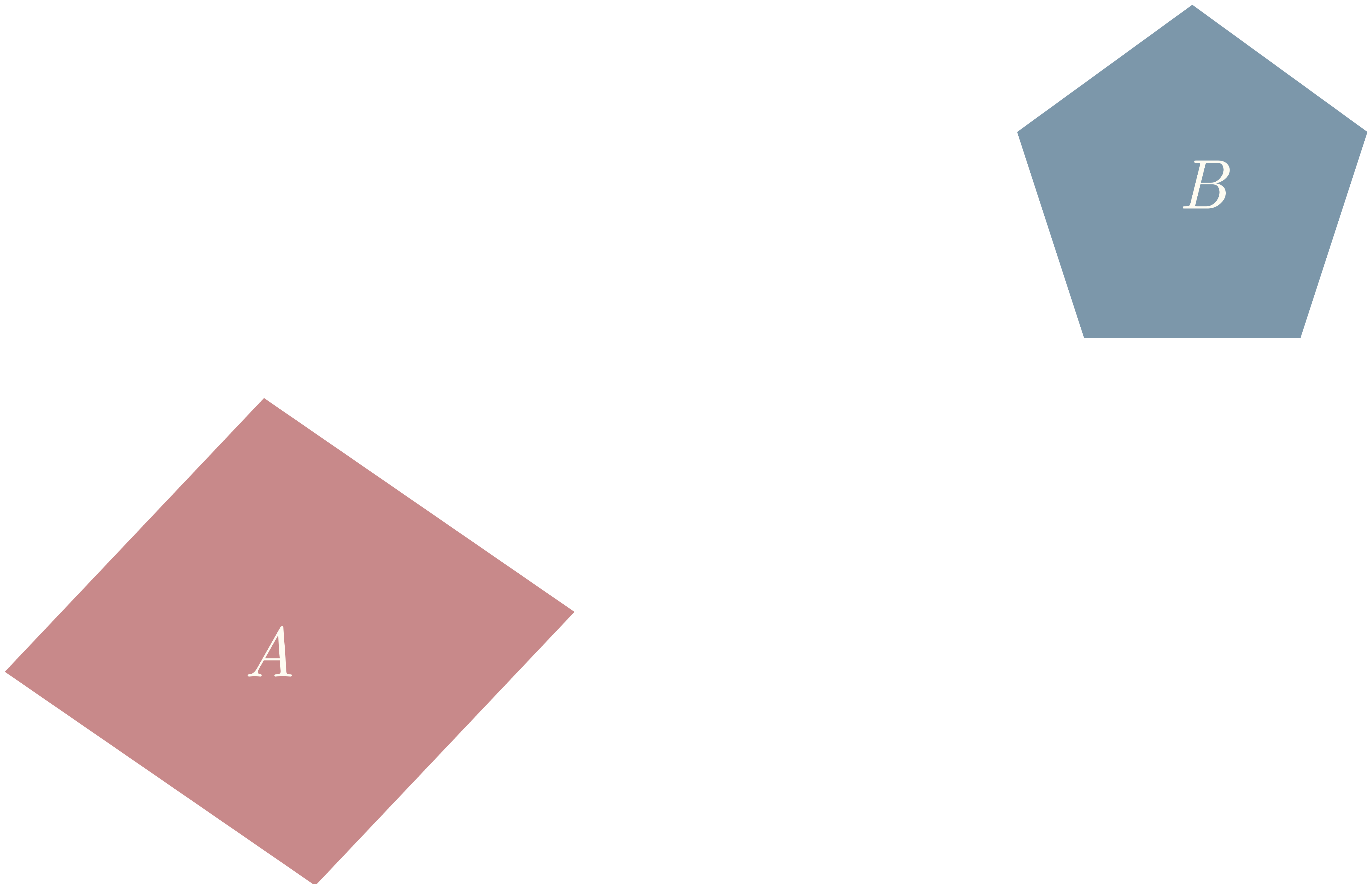
are also interpolants for $A$

$I \longrightarrow I'$

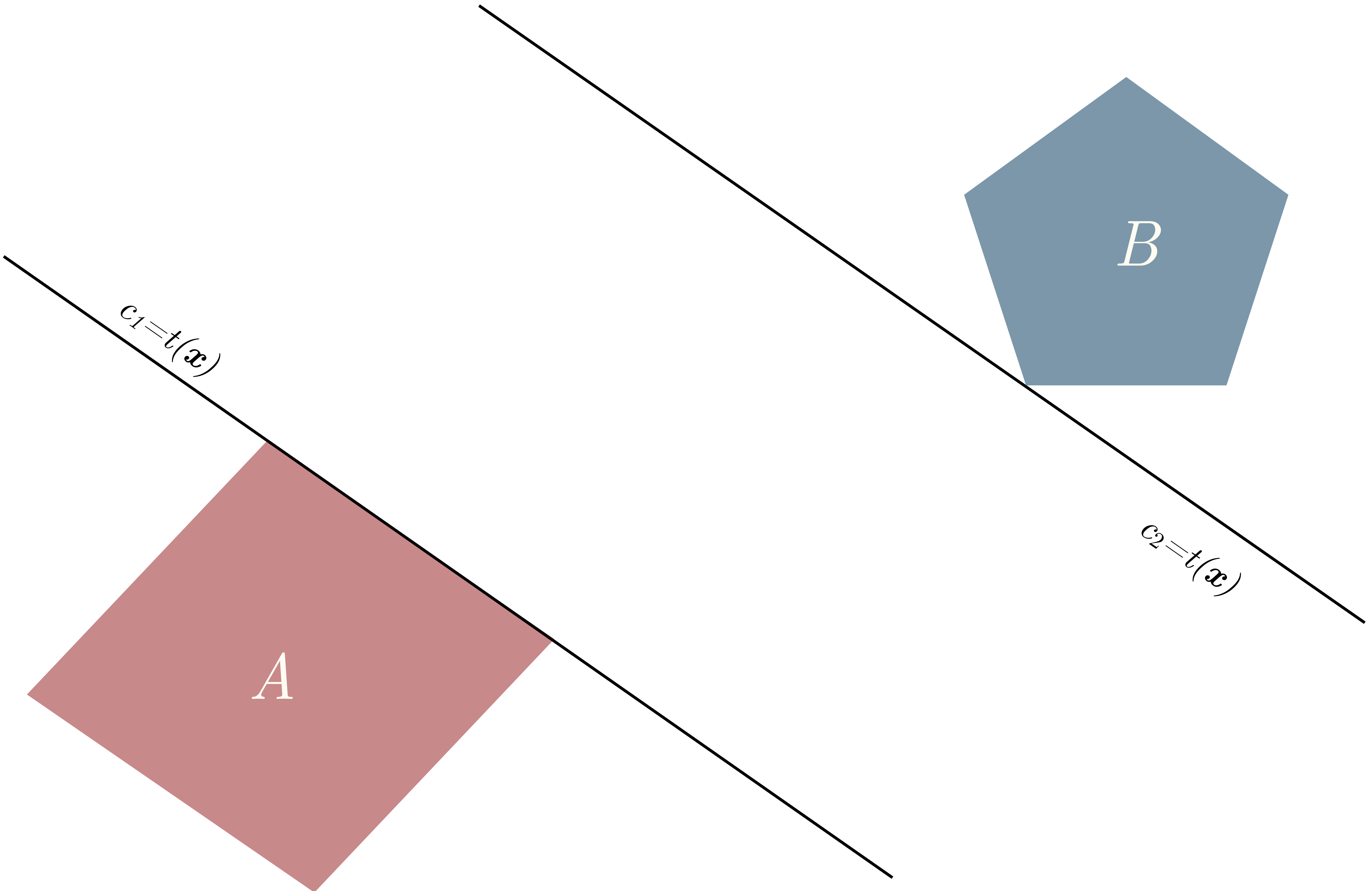
In the paper



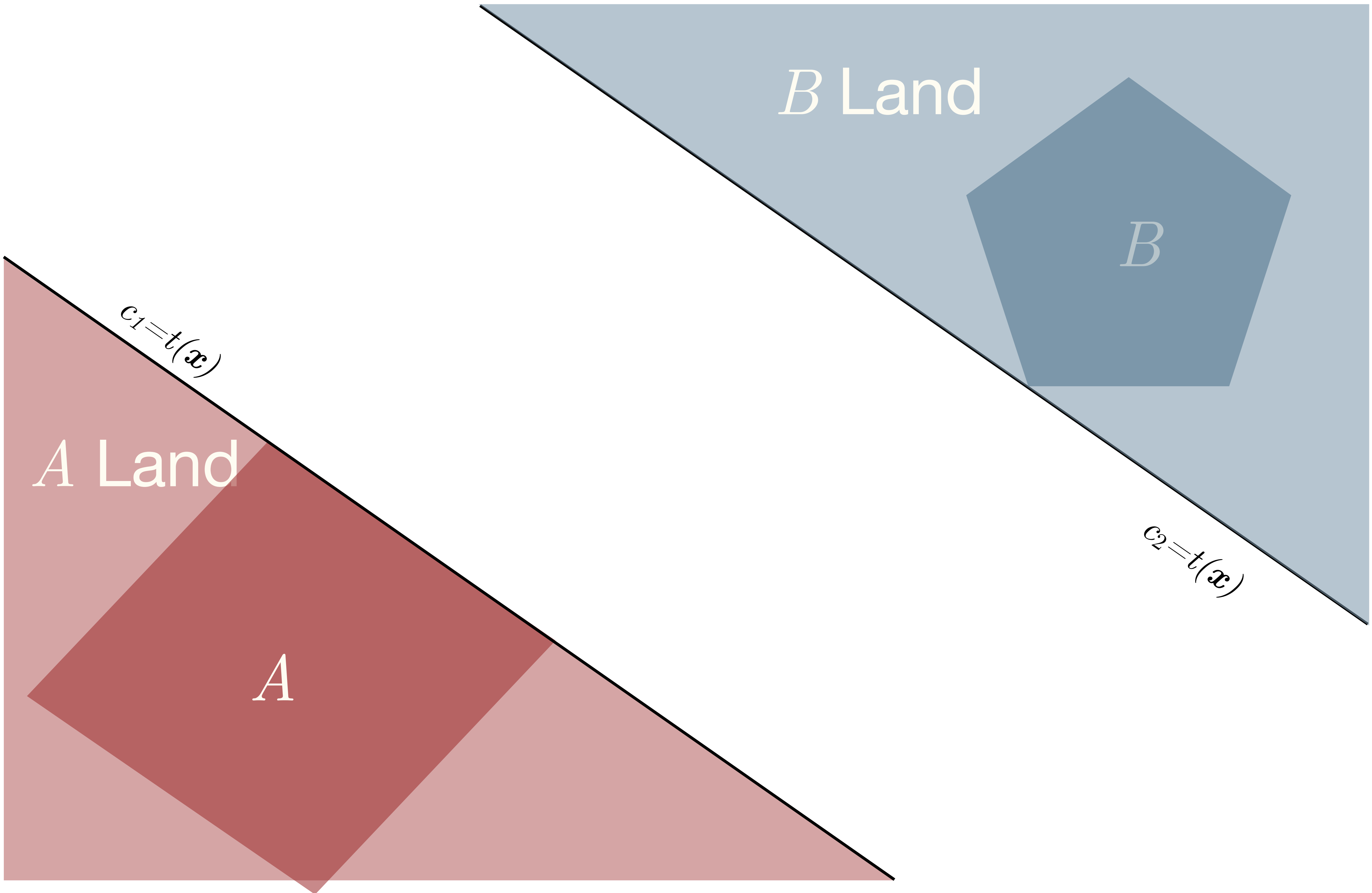
# Interpolant Duality Visualized



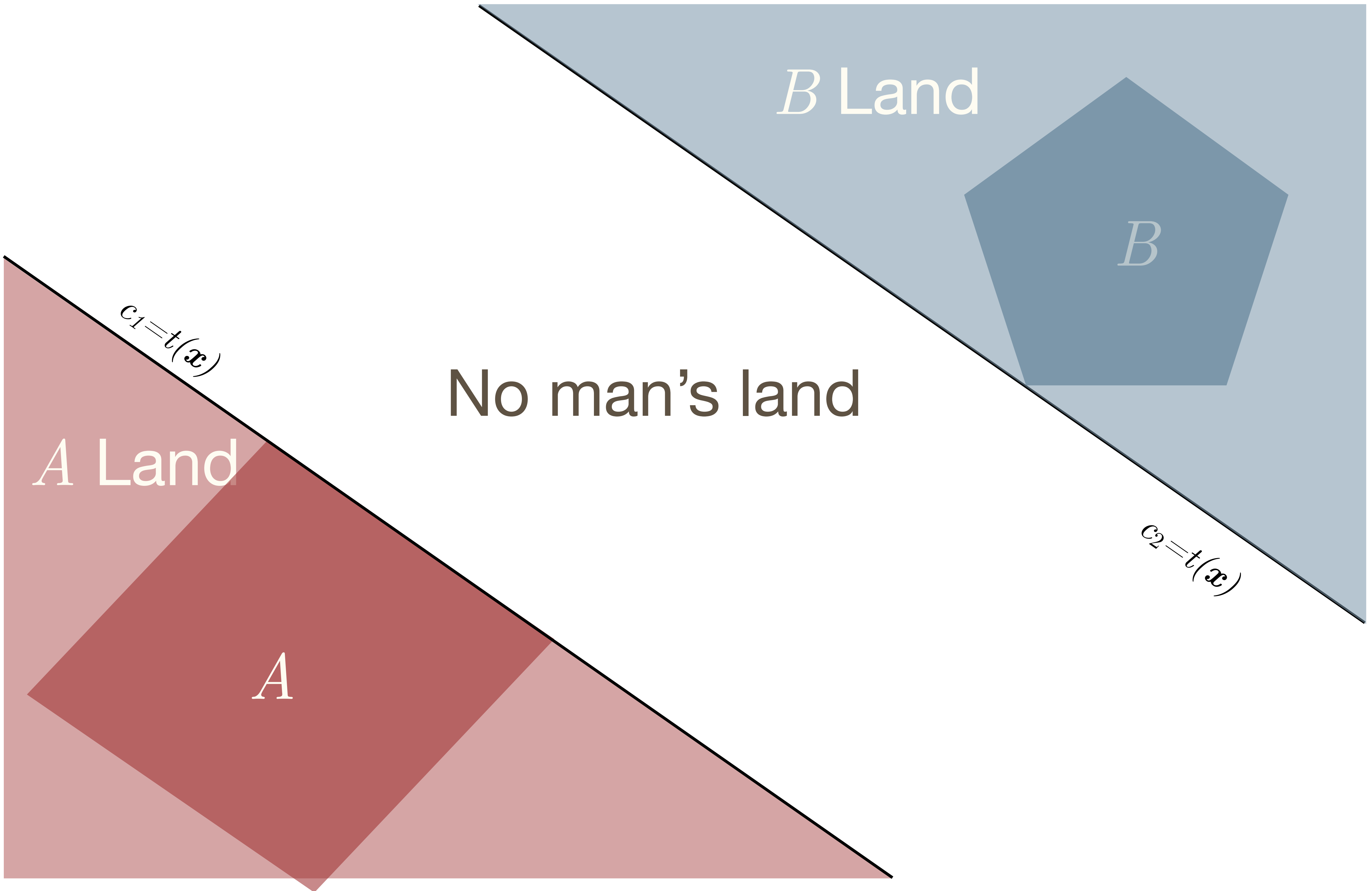
# Interpolant Duality Visualized



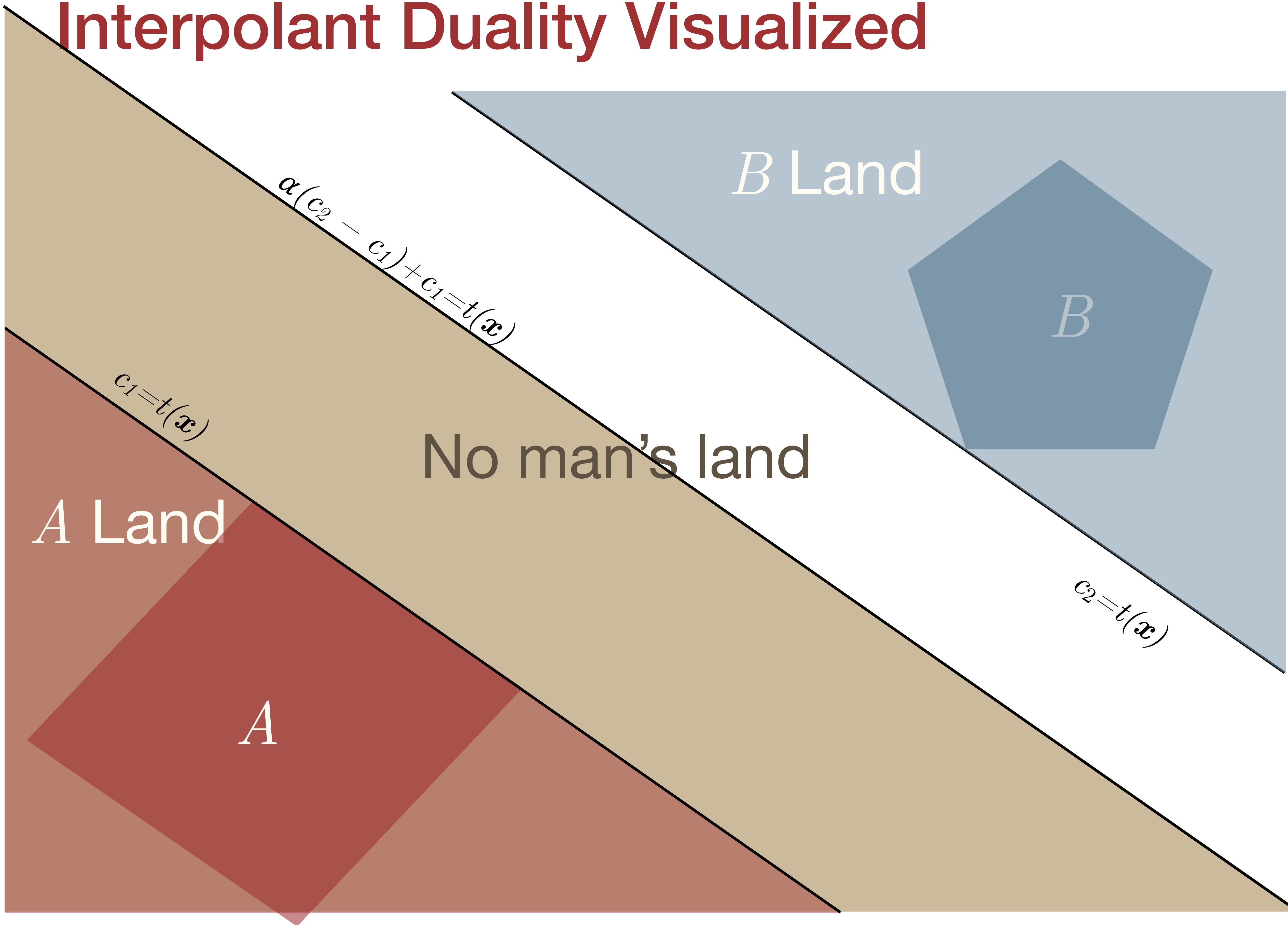
# Interpolant Duality Visualized



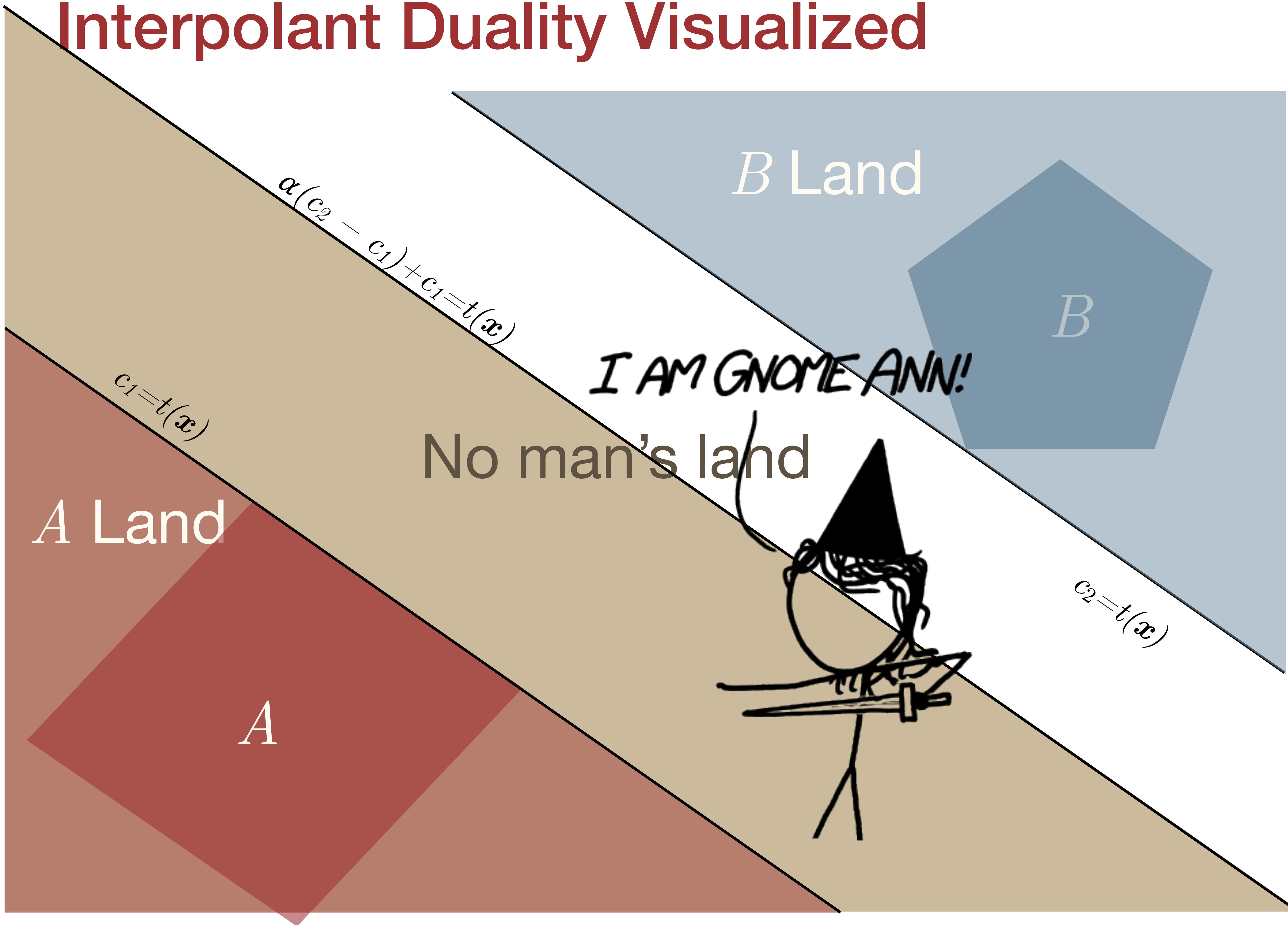
# Interpolant Duality Visualized



# Interpolant Duality Visualized

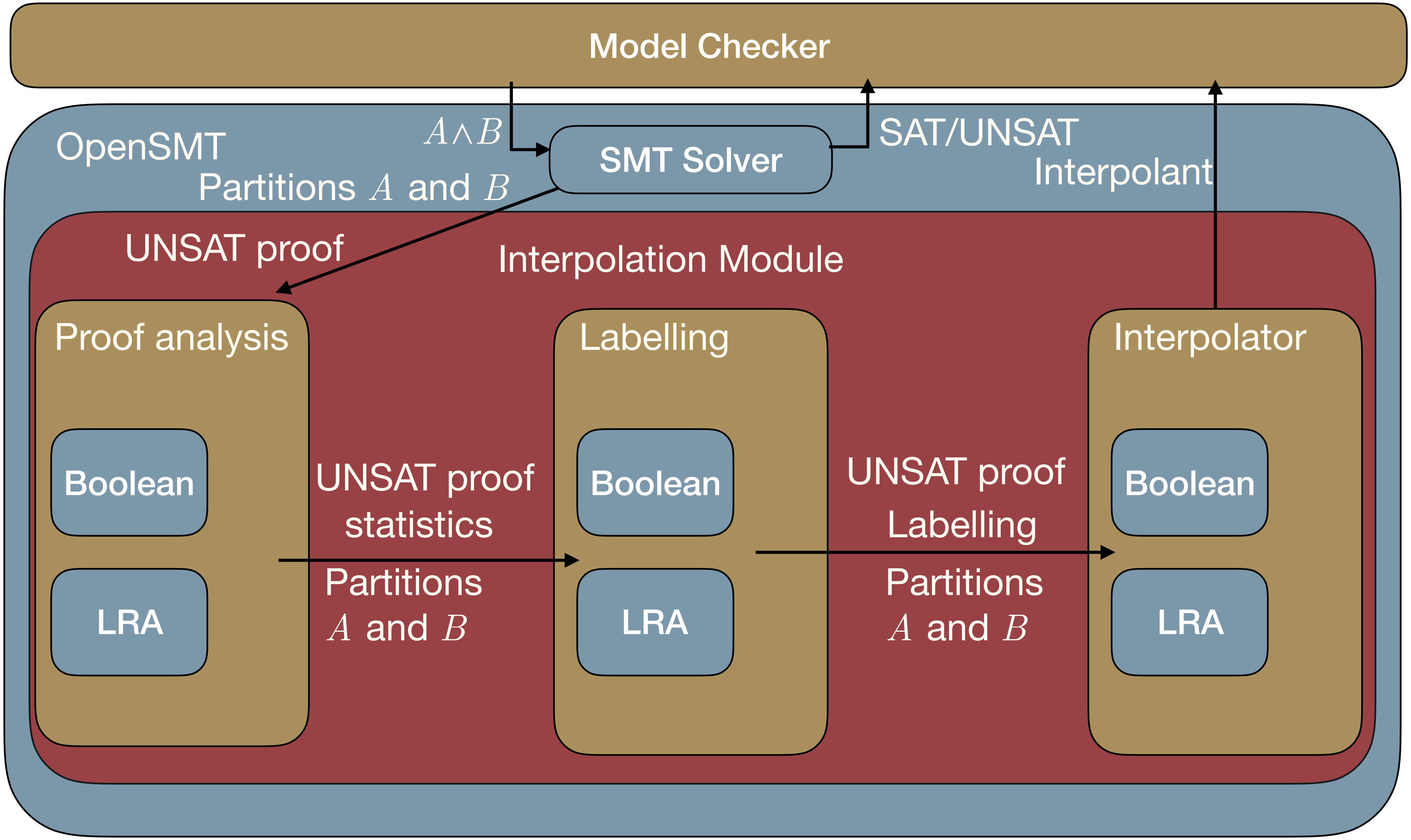


# Interpolant Duality Visualized

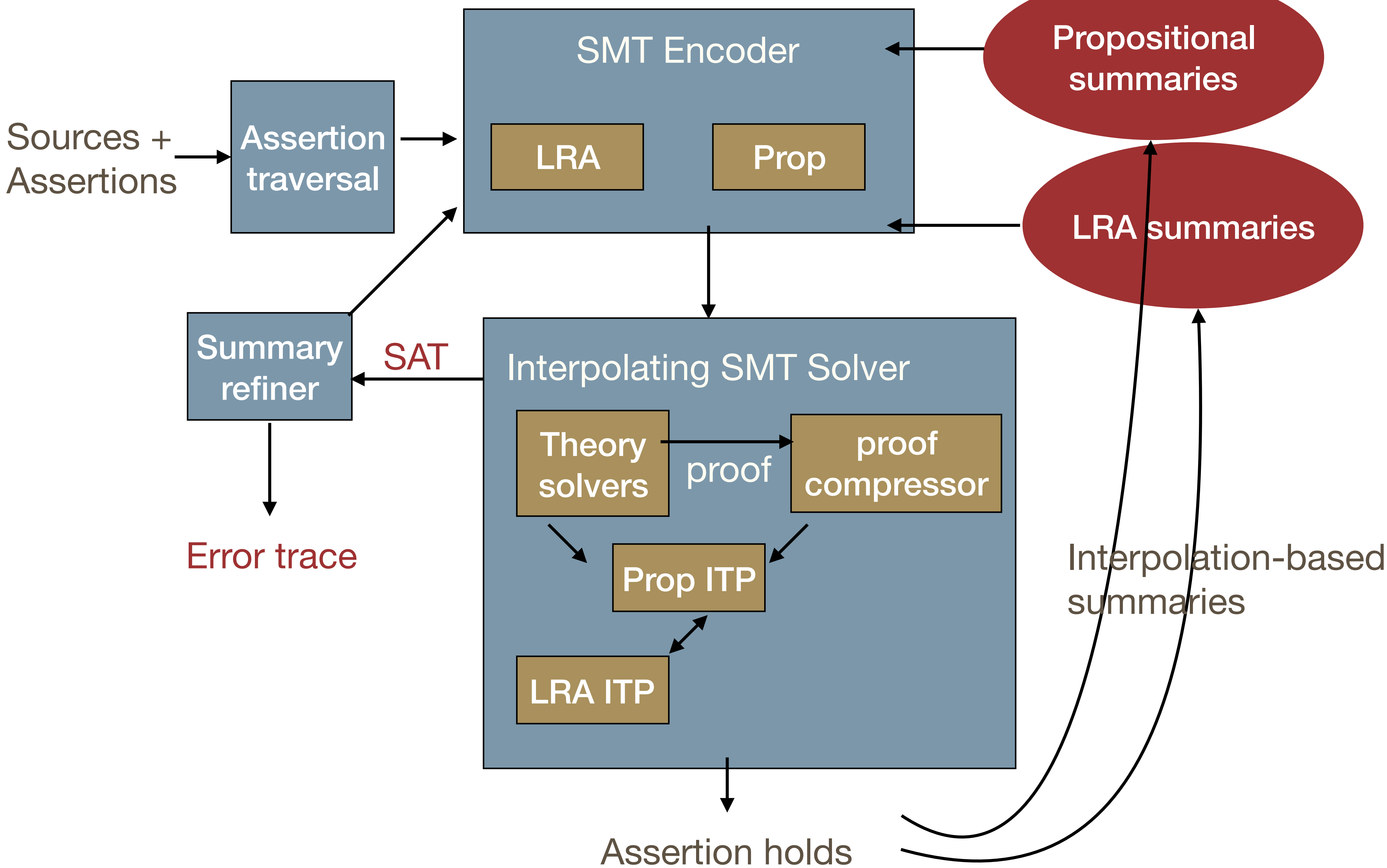


# Experiments on SV-COMP and HiFrog

# The Architecture Overview



# Implemented in HiFrog



# Results on SMT-LIB

Experiments with three LRA labelling functions:

Strong: the primal interpolant

Weak: the dual interpolant

$c = 0.5$: the interpolant between dual and primal

# Experiments on HiFrog

| ITP | floppy1 | kbfiltr1 | diskperf1 | mem | disk | Σ |
|---|---|---|---|---|---|---|
| **Strong** | 27100 | 5120 | 39900 | 25600 | 47600 | 145000 |
| $c = 0.5$ | 25100 | 5120 | 39200 | 25100 | 41500 | 136000 |
| **Weak** | 24800 | 5380 | 39200 | 25600 | 64000 | 159000 |

Number of HiFrog refinements (fixed propositional ITP algorithm)

- The difference between minimum and maximum is ~ 15%
- The $c = 0.5$ ITP provides the best results

# Related Work


Nikolaj Bjørner, Arie Gurfinkel:
*Property Directed Polyhedral Abstraction.* VMCAI 2015

Pudlák:
*Lower bounds for resolution and cutting plane proofs and monotone computations*. Journal of Symbolic Logic 1997.

McMillan:
*An Interpolating Theorem Prover*. Theoretical Computer Science 2005.

D'Silva, Kroening, Purandare, and Weissenbacher:
*Interpolant Strength*. VMCAI 2010.

Albarghouthi, McMillan:
*Beautiful interpolants. CAV 2013.*

Dutertre, de Moura:
*A fast linear-arithmetic solver for DPLL(T)*. Logical Methods in Computer Science 2012.

Alt, Hyvärinen, Asadi, and Sharygina:
*Duality-Based Interpolation for Quantifier-Free Equalities and Uninterpreted Functions. FMCAD 2017.*

# Conclusions

LRA interpolation with controlled strength

Provides an infinite family of interpolants based on interpolation duality

Integrated into a model checker

# Future work

Better heuristics for the labelling function

Apply to fix-point computations in other MC applications

Implementations available at

http://verify.inf.usi.ch/hifrog,

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