Stable model semantics for founded bounds
ICLP, 2013

Rehan Abdul Aziz, Geoffrey Chu, and Peter Stuckey

Computing and Information Systems
The University of Melbourne and
NICTA
raziz@student.unimelb.edu.au

August 14, 2013
Answer set solvers are more powerful than SAT solvers for solving a class of problems. These problems typically involve some kind of transitive closure calculations such as reachability.
Outline

Answer set solvers are more powerful than SAT solvers for solving a class of problems. These problems typically involve some kind of transitive closure calculations such as reachability.

ASP solvers do not natively support integer variables, they *ground* them to propositional variables.
Answer set solvers are more powerful than SAT solvers for solving a class of problems. These problems typically involve some kind of transitive closure calculations such as reachability.

ASP solvers do not natively support integer variables, they *ground* them to propositional variables.

The goal of this talk is to:
- Extend stable model semantics to integers without having to ground them.
Outline

Answer set solvers are more powerful than SAT solvers for solving a class of problems. These problems typically involve some kind of transitive closure calculations such as reachability.

ASP solvers do not natively support integer variables, they ground them to propositional variables.

The goal of this talk is to:

- Extend stable model semantics to integers without having to ground them.
- Describe an implementation in a CP solver.
Answer set solvers are more powerful than SAT solvers for solving a class of problems. These problems typically involve some kind of transitive closure calculations such as reachability.

ASP solvers do not natively support integer variables, they *ground* them to propositional variables.

The goal of this talk is to:

- Extend stable model semantics to integers without having to ground them.
- Describe an implementation in a CP solver.
Consider deciding which nodes are reachable from $a$: $a \quad b \quad c$
Consider deciding which nodes are reachable from $a$: $a \quad \overrightarrow{b} \quad c$

Let $r_X$ specify whether $X$ is reachable from $a$. An ASP model is:

$$\forall (X, Y) \in E \quad r_Y \leftarrow r_X.$$
Reachability (continued)

After grounding:

\[
\begin{align*}
& r_a. \\
& r_b \leftarrow r_c. \\
& r_c \leftarrow r_b.
\end{align*}
\]
Reachability (continued)

After grounding:

\[
\begin{align*}
& r_a. \\
& r_b \leftarrow r_c. \\
& r_c \leftarrow r_b.
\end{align*}
\]

A SAT solver would give the incorrect solution:
\[
r_a = r_b = r_c = true.
\]
Reachability (continued)

After grounding:

\[
\begin{align*}
& r_a. \\
& r_b \leftarrow r_c. \\
& r_c \leftarrow r_b.
\end{align*}
\]

A SAT solver would give the incorrect solution:

\[ r_a = r_b = r_c = true. \]

Modern ASP solvers are basically SAT solvers with an additional component to avoid this circular or unstable solution.
Reachability (continued)

After grounding:

\[
\begin{align*}
& r_a. \\
& r_b \leftarrow r_c. \\
& r_c \leftarrow r_b.
\end{align*}
\]

A SAT solver would give the incorrect solution:

\[r_a = r_b = r_c = true.\]

Modern ASP solvers are basically SAT solvers with an additional component to avoid this circular or unstable solution.

They will detect \(r_b = r_c = true\) as an unfounded set, a set of variables that make each other true without having any external reason to be true.
Shortest Path

Calculating shortest path in a graph is similar to calculating reachability.
Shortest Path

Calculating shortest path in a graph is similar to calculating reachability.

Like stable model semantics eliminates circular supports on reachability variables, we can do the same for upper bounds on shortest path variables.
Shortest Path

Calculating shortest path in a graph is similar to calculating reachability.

Like stable model semantics eliminates circular supports on reachability variables, we can do the same for *upper bounds* on shortest path variables. Here is a graph and an ASP model:

\[
\begin{align*}
  & \quad \downarrow^{10} \quad b \quad \uparrow^{2} \quad c \quad \downarrow^{2} \\
  & \quad a \quad \quad \quad b \quad \quad c
\end{align*}
\]

\[
spub_a(0)
\]
Shortest Path

Calculating shortest path in a graph is similar to calculating reachability.

Like stable model semantics eliminates circular supports on reachability variables, we can do the same for upper bounds on shortest path variables. Here is a graph and an ASP model:

\[
\begin{align*}
& a \quad 10 \quad b \quad 2 \quad c \\
\forall X & \quad spub_a(0) \\
\forall X & \quad spub_X(M)
\end{align*}
\]
Calculating shortest path in a graph is similar to calculating reachability.

Like stable model semantics eliminates circular supports on reachability variables, we can do the same for \textit{upper bounds} on shortest path variables. Here is a graph and an ASP model:

\[
\begin{align*}
\forall X & \quad spub_{a}(0) \\
\forall (X, Y) \in E, S \in 0 \ldots M & \quad spub_{X}(M)
\end{align*}
\]
Calculating shortest path in a graph is similar to calculating reachability.

Like stable model semantics eliminates circular supports on reachability variables, we can do the same for *upper bounds* on shortest path variables. Here is a graph and an ASP model:

$$a \xrightarrow{10} b \xleftrightarrow{2} c.$$  

$$\forall X \quad \forall (X, Y) \in E, S \in 0 \ldots M \quad spub_a(0) \quad spub_X(M) \quad spub_Y(S + L) \leftarrow spub_X(S) + e_{X,Y}(L)$$
After grounding all nodes and edges:

\[
\begin{align*}
&spub_a(0) \\
&spub_a(M), spub_b(M), spub_c(M) \\
&\forall S \in 0..M \quad spub_b(10 + S) \leftarrow spub_a(S)
\end{align*}
\]
Shortest path (continued)

After grounding all nodes and edges:

\[ spub_a(0) \]
\[ spub_a(M), spub_b(M), spub_c(M) \]
\[ \forall S \in 0..M \quad spub_b(10 + S) \leftarrow spub_a(S) \]
\[ \forall S \in 0..M \quad spub_b(2 + S) \leftarrow spub_c(S) \]
Shortest path (continued)

After grounding all nodes and edges:

\[
\begin{align*}
\forall S \in 0..M & \quad spub_b(10 + S) \leftarrow spub_a(S) \\
\forall S \in 0..M & \quad spub_b(2 + S) \leftarrow spub_c(S) \\
\forall S \in 0..M & \quad spub_c(2 + S) \leftarrow spub_b(S)
\end{align*}
\]
After grounding all nodes and edges:

\[
\begin{align*}
\forall S \in 0..M & \quad spub_b(10 + S) \leftarrow spub_a(S) \\
\forall S \in 0..M & \quad spub_b(2 + S) \leftarrow spub_c(S) \\
\forall S \in 0..M & \quad spub_c(2 + S) \leftarrow spub_b(S)
\end{align*}
\]

A SAT solver allows solutions where

\[
spub_a(0) = spub_b(0) = spub_c(0) = true \quad \text{which are false.}
\]
Grounding bottleneck

In this encoding, if $M$ is large the ground program can be huge.
Grounding bottleneck

In this encoding, if $M$ is large the ground program can be huge.

Constraint Answer Set Programming (CASP) systems cannot deal with such problems efficiently, since these systems do not have any notion of unfounded sets for integer or real variables.
Grounding bottleneck

In this encoding, if \( M \) is large the ground program can be huge.

Constraint Answer Set Programming (CASP) systems cannot deal with such problems efficiently, since these systems do not have any notion of unfounded sets for integer or real variables.

We need to detect unfounded sets on bounds of variables without grounding them. For this purpose, we extend stable model semantics first and unfounded sets to work for integer and reals.
Motivation

Grounding bottleneck

In this encoding, if $M$ is large the ground program can be huge.

Constraint Answer Set Programming (CASP) systems cannot deal with such problems efficiently, since these systems do not have any notion of unfounded sets for integer or real variables.

We need to detect unfounded sets on \textit{bounds} of variables \textit{without grounding them}. For this purpose, we extend stable model semantics first and unfounded sets to work for integer and reals.

If there is no rule for a variable, then it must default to lowest or highest value.
We introduce Bound Founded Answer Set Programming (BFASP). This system has two types of variables: **founded** ($\mathcal{F}$) and **standard** ($\mathcal{N}$) (CP/abstract/constraint) variables.
We introduce Bound Founded Answer Set Programming (BFASP). This system has two types of variables: *founded* ($\mathcal{F}$) and *standard* ($\mathcal{N}$) (CP/abstract/constraint) variables.

A *rule* can be *any* expression as long as it satisfies certain conditions. Formally, a rule is a pair ($c, y$) where $c$ is a constraint and $y \in \text{vars}(c) \cap \mathcal{F}$. 
We introduce Bound Founded Answer Set Programming (BFASP). This system has two types of variables: *founded* ($\mathcal{F}$) and *standard* ($\mathcal{N}$) (CP/abstract/constraint) variables.

A *rule* can be *any* expression as long as it satisfies certain conditions. Formally, a rule is a pair $(c, y)$ where $c$ is a constraint and $y \in \text{vars}(c) \cap \mathcal{F}$.

Constraints can be written as rules with no heads $(c, -)$. 
Founded variables can further be divided into lower and upper bound founded (lb-founded and ub-founded) variables. In absence of any rule, they default to \(-\infty\) and \(\infty\) respectively. To make them default to a different value, a bound can be added as a fact.
Lower and upper bound founded variables

Founded variables can further be divided into lower and upper bound founded (lb-founded and ub-founded) variables. In absence of any rule, they default to $-\infty$ and $\infty$ respectively. To make them default to a different value, a bound can be added as a fact.

ub-founded variables can be replaced by lb-founded variables.
Monotonicity

Key concept on which BFASP semantics is built.

**Definition**

An constraint $c$ is increasing (resp. decreasing) in its variable $x$ if increasing (resp. decreasing) the value of $x$ can never cause $c$ to go from satisfied to unsatisfied.
Monotonicity

Key concept on which BFASP semantics is built.

**Definition**

An constraint $c$ is increasing (resp. decreasing) in its variable $x$ if increasing (resp. decreasing) the value of $x$ can never cause $c$ to go from satisfied to unsatisfied.

**Example**

- $a \leftarrow b \land \neg c$ and $a \geq b - c$ are increasing in $a$, $c$ and decreasing in $b$. 
Monotonicity

Key concept on which BFASP semantics is built.

**Definition**

An constraint $c$ is increasing (resp. decreasing) in its variable $x$ if increasing (resp. decreasing) the value of $x$ can never cause $c$ to go from satisfied to unsatisfied.

**Example**

- $a \leftarrow b \land \neg c$ and $a \geq b - c$ are increasing in $a$, $c$ and decreasing in $b$.
- $a \geq |b|$ is non-monotonic in $b$. 
Monotonicity

Key concept on which BFASP semantics is built.

Definition

An constraint $c$ is increasing (resp. decreasing) in its variable $x$ if increasing (resp. decreasing) the value of $x$ can never cause $c$ to go from satisfied to unsatisfied.

Example

- $a \leftarrow b \land \neg c$ and $a \geq b - c$ are increasing in $a$, $c$ and decreasing in $b$.
- $a \geq |b|$ is non-monotonic in $b$. 
Definition

A HORN constraint is one which is increasing in at most one of its variables and decreasing in the rest. A HORN-CP is a collection of HORN constraints.
Definition

A HORN constraint is one which is increasing in at most one of its variables and decreasing in the rest. A HORN-CP is a collection of HORN constraints.

Example

Let $P = \{x \geq 0, y \geq 4 + x, 2x \geq y\}$. $P$ is a HORN-CP program, each constraint monotonically increasing in the first argument.
**Definition**

An assignment $\theta$ is the minimal assignment of a HORN-CP $P$ iff $\theta \models P$ and there is no other valuation $\theta'$ that also satisfies $P$ and $\theta'(v) < \theta(v)$ for some $v \in \text{vars}(P)$. 

**Example**

Let $P = \{x \geq 0, y \geq 4 + x, 2x \geq y\}$. A fixpoint calculation gives:

- $x \geq 0$
- $y \geq 4$
- $x \geq 2$
- $y \geq 6$
- $x \geq 3$
- $y \geq 7$
- $x \geq 4$
- $y \geq 8$

$\text{MA}(P) = \{x \mapsto 4, y \mapsto 8\}$. 

Every HORN-CP has at most one minimal assignment.
Minimal assignments

**Definition**

An assignment \( \theta \) is the minimal assignment of a HORN-CP \( P \) iff \( \theta \models P \) and there is no other valuation \( \theta' \) that also satisfies \( P \) and \( \theta'(v) < \theta(v) \) for some \( v \in \text{vars}(P) \).

**Example**

Let \( P = \{ x \geq 0, y \geq 4 + x, 2x \geq y \} \). A fixpoint calculation gives:

\[
x \geq 0,
y \geq 6,
x \geq 3,
y \geq 7,
x \geq 4,
y \geq 8.
\]

\( \text{MA}(P) = \{ x \mapsto 4, y \mapsto 8 \} \). Every HORN-CP has at most one minimal assignment.
Minimal assignments

Definition
An assignment $\theta$ is the minimal assignment of a HORN-CP $P$ iff $\theta \models P$ and there is no other valuation $\theta'$ that also satisfies $P$ and $\theta'(v) < \theta(v)$ for some $v \in \text{vars}(P)$.

Example
Let $P = \{x \geq 0, y \geq 4 + x, 2x \geq y\}$. A fixpoint calculation gives: $x \geq 0, y \geq 4$, $2x \geq y$. MA($P$) = $\{x \mapsto 4, y \mapsto 8\}$. Every HORN-CP has at most one minimal assignment.
**Definition**

An assignment $\theta$ is the minimal assignment of a HORN-CP $P$ iff $\theta \models P$ and there is no other valuation $\theta'$ that also satisfies $P$ and $\theta'(v) < \theta(v)$ for some $v \in \text{vars}(P)$.

**Example**

Let $P = \{x \geq 0, y \geq 4 + x, 2x \geq y\}$. A fixpoint calculation gives:

$x \geq 0, y \geq 4, x \geq 2,$
Minimal assignments

Definition
An assignment $\theta$ is the minimal assignment of a HORN-CP $P$ iff $\theta \models P$ and there is no other valuation $\theta'$ that also satisfies $P$ and $\theta'(v) < \theta(v)$ for some $v \in \text{vars}(P)$.

Example
Let $P = \{x \geq 0, y \geq 4 + x, 2x \geq y\}$. A fixpoint calculation gives:
$x \geq 0, y \geq 4, x \geq 2, y \geq 6,$
Minimal assignments

**Definition**
An assignment $\theta$ is the minimal assignment of a HORN-CP $P$ iff $\theta \models P$ and there is no other valuation $\theta'$ that also satisfies $P$ and $\theta'(v) < \theta(v)$ for some $v \in \text{vars}(P)$.

**Example**
Let $P = \{x \geq 0, y \geq 4 + x, 2x \geq y\}$. A fixpoint calculation gives: $x \geq 0, y \geq 4, x \geq 2, y \geq 6, x \geq 3$,
Minimal assignments

**Definition**

An assignment $\theta$ is the minimal assignment of a HORN-CP $P$ iff $\theta \models P$ and there is no other valuation $\theta'$ that also satisfies $P$ and $\theta'(v) < \theta(v)$ for some $v \in \text{vars}(P)$.

**Example**

Let $P = \{x \geq 0, y \geq 4 + x, 2x \geq y\}$. A fixpoint calculation gives:

$x \geq 0, y \geq 4, x \geq 2, y \geq 6, x \geq 3, y \geq 7,$
Minimal assignments

Definition

An assignment \( \theta \) is the minimal assignment of a HORN-CP \( P \) iff \( \theta \models P \) and there is no other valuation \( \theta' \) that also satisfies \( P \) and \( \theta'(v) < \theta(v) \) for some \( v \in \text{vars}(P) \).

Example

Let \( P = \{ x \geq 0, y \geq 4 + x, 2x \geq y \} \). A fixpoint calculation gives: \( x \geq 0, y \geq 4, x \geq 2, y \geq 6, x \geq 3, y \geq 7, x \geq 4 \),
Minimal assignments

**Definition**

An assignment $\theta$ is the minimal assignment of a HORN-CP $P$ iff $\theta \models P$ and there is no other valuation $\theta'$ that also satisfies $P$ and $\theta'(v) < \theta(v)$ for some $v \in \text{vars}(P)$.

**Example**

Let $P = \{x \geq 0, y \geq 4 + x, 2x \geq y\}$. A fixpoint calculation gives: $x \geq 0, y \geq 4, x \geq 2, y \geq 6, x \geq 3, y \geq 7, x \geq 4, y \geq 8$. 

Minimal assignments

**Definition**

An assignment $\theta$ is the minimal assignment of a HORN-CP $P$ iff $\theta \models P$ and there is no other valuation $\theta'$ that also satisfies $P$ and $\theta'(v) < \theta(v)$ for some $v \in \text{vars}(P)$.

**Example**

Let $P = \{ x \geq 0, y \geq 4 + x, 2x \geq y \}$. A fixpoint calculation gives: $x \geq 0, y \geq 4, x \geq 2, y \geq 6, x \geq 3, y \geq 7, x \geq 4, y \geq 8$. $MA(P) = \{ x \mapsto 4, y \mapsto 8 \}$.

Every HORN-CP has at most one minimal assignment.
Reduct

Definition

Given a BFASP $P$ and an assignment $\theta$, the CP-reduct of $P$ w.r.t. $\theta$ (written $P^\theta$) is a HORN-CP obtained by replacing in each rule $r = (c, y)$ every standard variable and every variable $v \in \text{vars}(c) \setminus \{y\}$ in which $c$ is not decreasing by $\theta(v)$. Discard the rule if it becomes a tautology.

Example

$P = \{(a \geq 7, a), (b \geq ad - c, b), (c \geq 2, c)\}$ where $a, b, c$ are lb-founded variables and $d$ is a standard variable.
Reduct

Definition

Given a BFASP \( P \) and an assignment \( \theta \), the CP-reduct of \( P \) w.r.t. \( \theta \) (written \( P^\theta \)) is a HORN-CP obtained by replacing in each rule \( r = (c, y) \) every standard variable and every variable \( v \in \text{vars}(c) \setminus \{y\} \) in which \( c \) is not decreasing by \( \theta(v) \). Discard the rule if it becomes a tautology.

Example

\( P = \{(a \geq 7, a), (b \geq ad - c, b), (c \geq 2, c)\} \) where \( a, b, c \) are \( \text{lb-founded variables} \) and \( d \) is a standard variable.

\[ \theta = \{a \mapsto 7, c \mapsto 2, d \mapsto 10, b \mapsto 68\} \]

\( P^\theta = \{a \geq 7, b \geq 10a - 2, c \geq 2\} \).
Stable assignments for BFASPs

**Definition**

θ is a stable assignment for a BFASP \( P \) iff \( \theta = MA(P^\theta) \) (restricted to founded variables).
Stable assignments for BFASPs

Definition

\( \theta \) is a stable assignment for a BFASP \( P \) iff \( \theta = \text{MA}(P^\theta) \) (restricted to founded variables).

Example

\( P = \{(a \geq 7, a), (b \geq ad - c, b), (c \geq 2, c)\} \) where \( a, b, c \) are \( lb \)-founded variables and \( d \) is a standard variable.

\( \theta = \{a \mapsto 7, c \mapsto 2, d \mapsto 10, b \mapsto 68\}, \)

\( P^\theta = \{a \geq 7, b \geq 10a - 2, c \geq 2\} \).
Stable assignments for BFASPs

**Definition**

\( \theta \) is a stable assignment for a BFASP \( P \) iff \( \theta = MA(P^\theta) \) (restricted to founded variables).

**Example**

\[ P = \{(a \geq 7, a), (b \geq ad - c, b), (c \geq 2, c)\} \] where \( a, b, c \) are \( lb \)-founded variables and \( d \) is a standard variable.

\[ \theta = \{a \mapsto 7, c \mapsto 2, d \mapsto 10, b \mapsto 68\}, \]

\[ P^\theta = \{a \geq 7, b \geq 10a - 2, c \geq 2\}. \]

\[ MA(P^\theta) =_F \theta, \text{ therefore, } \theta \text{ is a stable assignment}. \]
Here is a ground BFASP model, \( sp_x \) is a ub-founded variable representing the shortest path from \( a \) to \( x \):

\[
sp_a \leq 0
\]
Here is a ground BFASP model, $sp_X$ is a ub-founded variable representing the shortest path from $a$ to $x$:

\[
s_{pa} \leq 0 \\
s_{pb} \leq s_{pa} + 10
\]
Here is a ground BFASP model, \( sp_X \) is a ub-founded variable representing the shortest path from \( a \) to \( x \):

\[
\begin{align*}
sp_a & \leq 0 \\
sp_b & \leq sp_a + 10 \\
sp_b & \leq sp_c + 2
\end{align*}
\]
Here is a ground BFASP model, \( sp_X \) is a ub-founded variable representing the shortest path from \( a \) to \( x \):

\[
\begin{align*}
sp_a &\leq 0 \\
sp_b &\leq sp_a + 10 \\
sp_b &\leq sp_c + 2 \\
sp_c &\leq sp_b + 2
\end{align*}
\]
Implemented as a propagator in the constraint solver *chuffed* augmented with lb-founded variables and rules.
Implementation

Implemented as a propagator in the constraint solver chuffed augmented with lb-founded variables and rules.

Unfounded sets of bounds are detected during the search using the source pointer technique.
Overview of unfounded set detection

1. Maintain a *directed acyclic justification graph* of bounds that tell for every bound all other bound that were used to derive it.
Overview of unfounded set detection

1. Maintain a \textit{directed acyclic justification graph} of bounds that tell for every bound all other bound that were used to derive it.

2. If any bound becomes false due to a decision or propagation, remove or \textit{dejustify} all other bounds that can be reached from it.
Overview of unfounded set detection

1. Maintain a *directed acyclic justification graph* of bounds that tell for every bound all other bound that were used to derive it.

2. If any bound becomes false due to a decision or propagation, remove or *dejustify* all other bounds that can be reached from it.

3. Re-establish or *rejustify* all the dejustified bounds as much as possible and update the graph.
Maintain a *directed acyclic justification graph* of bounds that tell for every bound all other bound that were used to derive it.

If any bound becomes false due to a decision or propagation, remove or *dejustify* all other bounds that can be reached from it.

Re-establish or *rejustify* all the dejustified bounds as much as possible and update the graph.

The set of bounds that cannot be rejustified belong to some unfounded set, and must be set to false.
Overview of unfounded set detection

1. Maintain a *directed acyclic justification graph* of bounds that tell for every bound all other bound that were used to derive it.

2. If any bound becomes false due to a decision or propagation, remove or *dejustify* all other bounds that can be reached from it.

3. Re-establish or *rejustify* all the dejustified bounds as much as possible and update the graph.

4. The set of bounds that cannot be rejustified belong to some unfounded set, and must be set to false.

A justification is not the same as a decision.
Suppose a government wants to decide which policies to *enact* in order to maximize the happiness of its citizens. Given:
Utilitarian policies (example)

Suppose a government wants to decide which policies to *enact* in order to maximize the happiness of its citizens. Given:

- Cost of each policy and total available budget
Suppose a government wants to decide which policies to *enact* in order to maximize the happiness of its citizens. Given:

- Cost of each policy and total available budget
- Utility of each policy for each citizen
Utilitarian policies (example)

Suppose a government wants to decide which policies to *enact* in order to maximize the happiness of its citizens. Given:

- Cost of each policy and total available budget
- Utility of each policy for each citizen
- Effect of happiness of citizens upon each other’s happiness
Suppose a government wants to decide which policies to *enact* in order to maximize the happiness of its citizens. Given:

- Cost of each policy and total available budget
- Utility of each policy for each citizen
- Effect of happiness of citizens upon each other’s happiness

Decide which policies to enact such that the total happiness of citizens is maximized and the cost does not exceed the budget.
Consider the following utilitarian policies instance:

\[ R1 : h_1 \geq 5e_2 + 6(h_2 \geq 7) \quad R2 : h_2 \geq 9e_1 + 7(h_1 \geq 5) \]
Implementation

Consider the following utilitarian policies instance:

\[
R1 : h_1 \geq 5e_2 + 6(h_2 \geq 7) \quad R2 : h_2 \geq 9e_1 + 7(h_1 \geq 5)
\]

Without stable model semantics, we can have solutions like \( e_1 = e_2 = false, h_1 = 6, h_2 = 7 \), which do not make sense since no person was proven to be happy.
Implementation

\[ R1 : h_1 \geq 5e_2 + 6(h_2 \geq 7) \quad R2 : h_2 \geq 9e_1 + 7(h_1 \geq 5) \]
Implementation

\[ R1 : h_1 \geq 5e_2 + 6(h_2 \geq 7) \]
\[ R2 : h_2 \geq 9e_1 + 7(h_1 \geq 5) \]

Decision level 0: \( h_1 \geq -\infty, h_2 \geq -\infty \).
Implementation

\[ R1 : h_1 \geq 5e_2 + 6(h_2 \geq 7) \quad R2 : h_2 \geq 9e_1 + 7(h_1 \geq 5) \]

Decision level 0: \( h_1 \geq -\infty, \ h_2 \geq -\infty. \)

<table>
<thead>
<tr>
<th>Bound</th>
<th>Rule</th>
<th>Supports</th>
</tr>
</thead>
<tbody>
<tr>
<td>([h_1 \geq 5])</td>
<td>(R1)</td>
<td>([e_2 \neq \bot])</td>
</tr>
</tbody>
</table>
R1 : \( h_1 \geq 5e_2 + 6(h_2 \geq 7) \) \hspace{1cm} R2 : \( h_2 \geq 9e_1 + 7(h_1 \geq 5) \)

Decision level 0: \( h_1 \geq -\infty, \ h_2 \geq -\infty \).

<table>
<thead>
<tr>
<th>Bound</th>
<th>Rule</th>
<th>Supports</th>
</tr>
</thead>
<tbody>
<tr>
<td>( h_1 \geq 5 )</td>
<td>R1</td>
<td>( e_2 \neq \bot )</td>
</tr>
<tr>
<td>( h_2 \geq 9 )</td>
<td>R2</td>
<td>( e_1 \neq \bot )</td>
</tr>
</tbody>
</table>
Implementation

\[ R1 : h_1 \geq 5e_2 + 6(h_2 \geq 7) \quad R2 : h_2 \geq 9e_1 + 7(h_1 \geq 5) \]

Decision level 0: \( h_1 \geq -\infty, \ h_2 \geq -\infty \).

<table>
<thead>
<tr>
<th>Bound</th>
<th>Rule</th>
<th>Supports</th>
</tr>
</thead>
<tbody>
<tr>
<td>([h_1 \geq 5])</td>
<td>(R1)</td>
<td>([e_2 \neq \bot])</td>
</tr>
<tr>
<td>([h_2 \geq 9])</td>
<td>(R2)</td>
<td>([e_1 \neq \bot])</td>
</tr>
<tr>
<td>([h_1 \geq 11])</td>
<td>(R1)</td>
<td>([e_2 \neq \bot], [h_2 \geq 7])</td>
</tr>
</tbody>
</table>
Implementation

\[ R1 : h_1 \geq 5e_2 + 6(h_2 \geq 7) \quad R2 : h_2 \geq 9e_1 + 7(h_1 \geq 5) \]

Decision level 0: \( h_1 \geq -\infty, \ h_2 \geq -\infty \).

<table>
<thead>
<tr>
<th>Bound</th>
<th>Rule</th>
<th>Supports</th>
</tr>
</thead>
<tbody>
<tr>
<td>([h_1 \geq 5])</td>
<td>(R1)</td>
<td>([e_2 \neq \bot])</td>
</tr>
<tr>
<td>([h_2 \geq 9])</td>
<td>(R2)</td>
<td>([e_1 \neq \bot])</td>
</tr>
<tr>
<td>([h_1 \geq 11])</td>
<td>(R1)</td>
<td>([e_2 \neq \bot], [h_2 \geq 7])</td>
</tr>
<tr>
<td>([h_2 \geq 16])</td>
<td>(R2)</td>
<td>([e_1 \neq \bot], [h_1 \geq 5])</td>
</tr>
</tbody>
</table>

We propagate \([h_1 \leq 11]\) and \([h_2 \leq 16]\).
Implementation

\[ R1 : h_1 \geq 5e_2 + 6(h_2 \geq 7) \quad R2 : h_2 \geq 9e_1 + 7(h_1 \geq 5) \]

Decision level 0: \( h_1 \geq -\infty, h_2 \geq -\infty \).

<table>
<thead>
<tr>
<th>Bound</th>
<th>Rule</th>
<th>Supports</th>
</tr>
</thead>
<tbody>
<tr>
<td>([h_1 \geq 5])</td>
<td>(R1)</td>
<td>([e_2 \neq \bot])</td>
</tr>
<tr>
<td>([h_2 \geq 9])</td>
<td>(R2)</td>
<td>([e_1 \neq \bot])</td>
</tr>
<tr>
<td>([h_1 \geq 11])</td>
<td>(R1)</td>
<td>([e_2 \neq \bot], [h_2 \geq 7])</td>
</tr>
<tr>
<td>([h_2 \geq 16])</td>
<td>(R2)</td>
<td>([e_1 \neq \bot], [h_1 \geq 5])</td>
</tr>
</tbody>
</table>

We propagate \([h_1 \leq 11]\) and \([h_2 \leq 16]\).
Implementation

\[ R1 : h_1 \geq 5e_2 + 6(h_2 \geq 7) \quad R2 : h_2 \geq 9e_1 + 7(h_1 \geq 5) \]

Decision level 0: \( h_1 \geq -\infty, \ h_2 \geq -\infty \).
Decision level 1: \([e_1 = \bot]\).

<table>
<thead>
<tr>
<th>Bound</th>
<th>Rule</th>
<th>Supports</th>
</tr>
</thead>
<tbody>
<tr>
<td>(h_1 \geq 5)</td>
<td>(R1)</td>
<td>([e_2 \neq \bot])</td>
</tr>
<tr>
<td>(h_2 \geq 9)</td>
<td>(R2)</td>
<td>([e_1 \neq \bot])</td>
</tr>
<tr>
<td>(h_1 \geq 11)</td>
<td>(R1)</td>
<td>([e_2 \neq \bot], [h_2 \geq 7])</td>
</tr>
<tr>
<td>(h_2 \geq 16)</td>
<td>(R2)</td>
<td>([e_1 \neq \bot], [h_1 \geq 5])</td>
</tr>
</tbody>
</table>
Implementation (Dejustification)

\[ R1 : h_1 \geq 5e_2 + 6(h_2 \geq 7) \quad R2 : h_2 \geq 9e_1 + 7(h_1 \geq 5) \]

Decision level 0: \( h_1 \geq -\infty, \ h_2 \geq -\infty \).
Decision level 1: \([e_1 = \bot]\).

<table>
<thead>
<tr>
<th>Bound</th>
<th>Rule</th>
<th>Supports</th>
</tr>
</thead>
<tbody>
<tr>
<td>([h_1 \geq 5])</td>
<td>(R1)</td>
<td>([e_2 \neq \bot])</td>
</tr>
<tr>
<td>([h_1 \geq 11])</td>
<td>(R1)</td>
<td>([e_2 \neq \bot], [h_2 \geq 7])</td>
</tr>
<tr>
<td>([h_2 \geq 16])</td>
<td>(R2)</td>
<td>([e_1 \neq \bot], [h_1 \geq 5])</td>
</tr>
</tbody>
</table>
Implementation (Dejustification)

\[ R1 : h_1 \geq 5e_2 + 6(h_2 \geq 7) \quad R2 : h_2 \geq 9e_1 + 7(h_1 \geq 5) \]

Decision level 0: \( h_1 \geq -\infty, \ h_2 \geq -\infty \).

Decision level 1: \( [e_1 = \bot] \).

<table>
<thead>
<tr>
<th>Bound</th>
<th>Rule</th>
<th>Supports</th>
</tr>
</thead>
<tbody>
<tr>
<td>([h_1 \geq 5])</td>
<td>(R1)</td>
<td>([e_2 \neq \bot])</td>
</tr>
<tr>
<td>([h_1 \geq 11])</td>
<td>(R1)</td>
<td>([e_2 \neq \bot], [h_2 \geq 7])</td>
</tr>
</tbody>
</table>
Implementation (Dejustification)

\[ R1 : h_1 \geq 5e_2 + 6(h_2 \geq 7) \quad R2 : h_2 \geq 9e_1 + 7(h_1 \geq 5) \]

Decision level 0: \( h_1 \geq -\infty, \quad h_2 \geq -\infty \).

Decision level 1: \([e_1 = \bot]\).

<table>
<thead>
<tr>
<th>Bound</th>
<th>Rule</th>
<th>Supports</th>
</tr>
</thead>
<tbody>
<tr>
<td>([h_1 \geq 5])</td>
<td>(R1)</td>
<td>([e_2 \neq \bot])</td>
</tr>
</tbody>
</table>
Implementation (Rejustification)

\[ R1 : h_1 \geq 5e_2 + 6(h_2 \geq 7) \quad R2 : h_2 \geq 9e_1 + 7(h_1 \geq 5) \]

Decision level 0: \( h_1 \geq -\infty, \ h_2 \geq -\infty \).
Decision level 1: \([e_1 = \bot]\).

<table>
<thead>
<tr>
<th>Bound</th>
<th>Rule</th>
<th>Supports</th>
</tr>
</thead>
<tbody>
<tr>
<td>([h_1 \geq 5])</td>
<td>(R1)</td>
<td>([e_2 \neq \bot])</td>
</tr>
<tr>
<td>([h_2 \geq 7])</td>
<td>(R2)</td>
<td>([h_1 \geq 5])</td>
</tr>
</tbody>
</table>
Implementation (Rejustification)

\[ R1 : h_1 \geq 5e_2 + 6(h_2 \geq 7) \quad R2 : h_2 \geq 9e_1 + 7(h_1 \geq 5) \]

Decision level 0: \( h_1 \geq -\infty, \ h_2 \geq -\infty \).

Decision level 1: \([e_1 = \bot]\).

<table>
<thead>
<tr>
<th>Bound</th>
<th>Rule</th>
<th>Supports</th>
</tr>
</thead>
<tbody>
<tr>
<td>([h_1 \geq 5])</td>
<td>(R1)</td>
<td>([e_2 \neq \bot])</td>
</tr>
<tr>
<td>([h_2 \geq 7])</td>
<td>(R2)</td>
<td>([h_1 \geq 5])</td>
</tr>
<tr>
<td>([h_1 \geq 11])</td>
<td>(R1)</td>
<td>([e_2 \neq \bot], [h_2 \geq 7])</td>
</tr>
</tbody>
</table>
Stable model semantics for founded bounds

Implementation

Implementation (Rejustification)

\[ R1 : h_1 \geq 5e_2 + 6(h_2 \geq 7) \quad R2 : h_2 \geq 9e_1 + 7(h_1 \geq 5) \]

Decision level 0: \( h_1 \geq -\infty, h_2 \geq -\infty \).
Decision level 1: \( e_1 = \bot \).

<table>
<thead>
<tr>
<th>Bound</th>
<th>Rule</th>
<th>Supports</th>
</tr>
</thead>
<tbody>
<tr>
<td>([h_1 \geq 5])</td>
<td>(R1)</td>
<td>([e_2 \neq \bot])</td>
</tr>
<tr>
<td>([h_2 \geq 7])</td>
<td>(R2)</td>
<td>([h_1 \geq 5])</td>
</tr>
<tr>
<td>([h_1 \geq 11])</td>
<td>(R1)</td>
<td>([e_2 \neq \bot], [h_2 \geq 7])</td>
</tr>
</tbody>
</table>

We propagate \( h_2 \leq 7 \leftarrow e_1 = \bot \).
Implementation

\[
R1 : h_1 \geq 5e_2 + 6(h_2 \geq 7) \quad R2 : h_2 \geq 9e_1 + 7(h_1 \geq 5)
\]

Decision level 0: \( h_1 \geq -\infty, \ h_2 \geq -\infty \).
Decision level 1: \( [e_1 = \bot] \).
Decision level 2: \( [e_2 = \bot] \).

<table>
<thead>
<tr>
<th>Bound</th>
<th>Rule</th>
<th>Supports</th>
</tr>
</thead>
<tbody>
<tr>
<td>([h_1 \geq 5])</td>
<td>(R1)</td>
<td>([e_2 \neq \bot])</td>
</tr>
<tr>
<td>([h_2 \geq 7])</td>
<td>(R2)</td>
<td>([h_1 \geq 5])</td>
</tr>
<tr>
<td>([h_1 \geq 11])</td>
<td>(R1)</td>
<td>([e_2 \neq \bot], [h_2 \geq 7])</td>
</tr>
</tbody>
</table>
Implementation (Dejustification)

\[ R1 : h_1 \geq 5e_2 + 6(h_2 \geq 7) \quad R2 : h_2 \geq 9e_1 + 7(h_1 \geq 5) \]

Decision level 0: \( h_1 \geq -\infty, \ h_2 \geq -\infty \).
Decision level 1: \([e_1 = \bot]\).
Decision level 2: \([e_2 = \bot]\).

<table>
<thead>
<tr>
<th>Bound</th>
<th>Rule</th>
<th>Supports</th>
</tr>
</thead>
<tbody>
<tr>
<td>([h_2 \geq 7])</td>
<td>(R2)</td>
<td>([h_1 \geq 5])</td>
</tr>
<tr>
<td>([h_1 \geq 11])</td>
<td>(R1)</td>
<td>([e_2 \neq \bot]), ([h_2 \geq 7])</td>
</tr>
</tbody>
</table>
R1 : $h_1 \geq 5e_2 + 6(h_2 \geq 7)$  \hspace{1cm} R2 : $h_2 \geq 9e_1 + 7(h_1 \geq 5)$

Decision level 0: $h_1 \geq -\infty$, $h_2 \geq -\infty$.
Decision level 1: $[e_1 = \bot]$.
Decision level 2: $[e_2 = \bot]$.

Bound  \hspace{1cm} Rule  \hspace{1cm} Supports

$[h_2 \geq 7]$  \hspace{1cm} R2  \hspace{1cm} $[h_1 \geq 5]$
Implementation (Dejustification)

\[ R1 : h_1 \geq 5e_2 + 6(h_2 \geq 7) \quad R2 : h_2 \geq 9e_1 + 7(h_1 \geq 5) \]

Decision level 0: \( h_1 \geq -\infty, \ h_2 \geq -\infty \).
Decision level 1: \( [e_1 = \bot] \).
Decision level 2: \( [e_2 = \bot] \).
Implementation (Dejustification)

\[ R_1 : h_1 \geq 5e_2 + 6(h_2 \geq 7) \quad R_2 : h_2 \geq 9e_1 + 7(h_1 \geq 5) \]

Decision level 0: \( h_1 \geq -\infty, \ h_2 \geq -\infty \).

Decision level 1: \([e_1 = \bot]\).

Decision level 2: \([e_2 = \bot]\).

No rejustification, we propagate \([h_1 \leq -\infty] \leftarrow [e_1 = \bot], [e_2 = \bot]\)
and \([h_2 \leq -\infty] \leftarrow [e_1 = \bot], [e_2 = \bot]\).
Experiments

Four benchmarks, all involving founded integers:
- Shortest path
Experiments

Four benchmarks, all involving founded integers:
- Shortest path
- Road construction
Four benchmarks, all involving founded integers:

- Shortest path
- Road construction
- Company controls (large number of stocks)
Experiments

Four benchmarks, all involving founded integers:

- Shortest path
- Road construction
- Company controls (large number of stocks)
- Utilitarian policies
### Experiments

<table>
<thead>
<tr>
<th>$N$</th>
<th>$E$</th>
<th>cl+gr</th>
<th>chuffed</th>
<th>cpx</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>gr</td>
<td>flat</td>
<td>flat</td>
</tr>
<tr>
<td></td>
<td></td>
<td>comp</td>
<td>comp</td>
<td>comp</td>
</tr>
<tr>
<td></td>
<td></td>
<td>time</td>
<td>time</td>
<td>time</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>6.00</td>
<td>0.01</td>
<td>0.03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.66</td>
<td>0.00</td>
<td>0.06</td>
</tr>
<tr>
<td>50</td>
<td>20</td>
<td>—</td>
<td>0.05</td>
<td>0.37</td>
</tr>
<tr>
<td></td>
<td></td>
<td>—</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td>—</td>
<td>0.01</td>
<td>0.36</td>
</tr>
<tr>
<td>100</td>
<td>30</td>
<td>—</td>
<td>0.25</td>
<td>3.12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>—</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td></td>
<td>—</td>
<td>0.12</td>
<td>600</td>
</tr>
<tr>
<td>250</td>
<td>100</td>
<td>—</td>
<td>1.00</td>
<td>28.11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>—</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td></td>
<td>—</td>
<td>1.16</td>
<td>600</td>
</tr>
<tr>
<td>600</td>
<td>200</td>
<td>—</td>
<td>2.50</td>
<td>169.08</td>
</tr>
<tr>
<td></td>
<td></td>
<td>—</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td></td>
<td>—</td>
<td>8.97</td>
<td>600</td>
</tr>
<tr>
<td>800</td>
<td>200</td>
<td>—</td>
<td>12.79</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td>—</td>
<td>N</td>
<td>600</td>
</tr>
<tr>
<td></td>
<td></td>
<td>—</td>
<td>600</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

**Table:** *Shortest path*
### Experiments

<table>
<thead>
<tr>
<th>$N$</th>
<th>$D$</th>
<th>$C$</th>
<th>cl+gr</th>
<th>gr</th>
<th>opt</th>
<th>comp</th>
<th>time</th>
<th>flat</th>
<th>opt</th>
<th>comp</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>10</td>
<td>8</td>
<td>0.43</td>
<td>24</td>
<td>Y</td>
<td>0.65</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>11</td>
<td>13</td>
<td>111.79</td>
<td>—</td>
<td>N</td>
<td>600</td>
<td>0.02</td>
<td>123</td>
<td>Y</td>
<td>0.06</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>45</td>
<td>40</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>0.66</td>
<td>2629</td>
<td>Y</td>
<td>3.46</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>25</td>
<td>32</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>4.91</td>
<td>2174531</td>
<td>N</td>
<td>600</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>30</td>
<td>25</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>13.15</td>
<td>6613781</td>
<td>N</td>
<td>600</td>
<td></td>
</tr>
</tbody>
</table>

**Table:** Road construction
### Experiments

<table>
<thead>
<tr>
<th>$S$</th>
<th>$C$</th>
<th>$cl+gr$</th>
<th>$opt$</th>
<th>$comp$</th>
<th>$time$</th>
<th>$chuffed$</th>
<th>$flat$</th>
<th>$opt$</th>
<th>$comp$</th>
<th>$time$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>5</td>
<td>0.00</td>
<td>102</td>
<td>Y</td>
<td>0.00</td>
<td>0.00</td>
<td>102</td>
<td>Y</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>50</td>
<td>10</td>
<td>0.05</td>
<td>702</td>
<td>N</td>
<td>600</td>
<td>0.02</td>
<td>234</td>
<td>Y</td>
<td>0.14</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>30</td>
<td>0.97</td>
<td>14701</td>
<td>N</td>
<td>600</td>
<td>0.22</td>
<td>408</td>
<td>Y</td>
<td>5.99</td>
<td></td>
</tr>
<tr>
<td>250</td>
<td>30</td>
<td>2.65</td>
<td>—</td>
<td>N</td>
<td>600</td>
<td>0.21</td>
<td>75</td>
<td>Y</td>
<td>35.52</td>
<td></td>
</tr>
<tr>
<td>500</td>
<td>50</td>
<td>16.63</td>
<td>—</td>
<td>N</td>
<td>600</td>
<td>0.72</td>
<td>—</td>
<td>N</td>
<td>600</td>
<td></td>
</tr>
</tbody>
</table>

**Table:** *Company controls*
### Experiments

<table>
<thead>
<tr>
<th>C</th>
<th>P</th>
<th>cl+gr</th>
<th>chuffed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>gr</td>
<td>opt</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>.07</td>
<td>34</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>66.64</td>
<td>—</td>
</tr>
<tr>
<td>20</td>
<td>25</td>
<td>89.10</td>
<td>—</td>
</tr>
<tr>
<td>90</td>
<td>90</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>250</td>
<td>230</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

**Table**: *Utilitarian policies*
### Experiments

<table>
<thead>
<tr>
<th>Scale</th>
<th>gr kB</th>
<th>cl+gr</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>748</td>
<td>0.36</td>
</tr>
<tr>
<td>2</td>
<td>2816</td>
<td>1.25</td>
</tr>
<tr>
<td>4</td>
<td>18741</td>
<td>37.83</td>
</tr>
<tr>
<td>6</td>
<td>67570</td>
<td>260.39</td>
</tr>
<tr>
<td>8</td>
<td>170287</td>
<td>600</td>
</tr>
<tr>
<td>10</td>
<td>336386</td>
<td>600</td>
</tr>
</tbody>
</table>

**Table:** *Utilitarian policies:* scaling behaviour on the smallest instance

chuffed requires 3.6kB and less than 0.005 seconds for all instances.
Conclusion

- BFASPs can solve a much wider range of problems as compared to any ASP, CP, or CASP system.
Conclusion

- BFASPs can solve a much wider range of problems as compared to any ASP, CP, or CASP system.
- Steals the closed world assumption of ASP and generalizes it for CP.
Conclusion

- BFASPs can solve a much wider range of problems as compared to any ASP, CP, or CASP system.
- *Steals* the closed world assumption of ASP and generalizes it for CP.
- A very general framework that makes it easier to add semantics for new rule forms.
BFASPs can solve a much wider range of problems as compared to any ASP, CP, or CASP system.

*Steals* the closed world assumption of ASP and generalizes it for CP.

A very general framework that makes it easier to add semantics for new rule forms.

Implemented in state of the art lazy clause generator. Therefore, BFASP = best of CP+SAT+ASP.
Questions

Thank you!
Some examples of founded quantities from modelling perspective are:

- ASP variables, default to \textit{false}. (lb-founded Booleans)
Some examples of founded quantities from modelling perspective are:

- ASP variables, default to *false*. (lb-founded Booleans)
- Concepts like trust, reputation, happiness etc. which need a reason to go up and by default might not exist. (lb-founded quantities)
Lower and upper bound founded variables

Some examples of founded quantities from modelling perspective are:

- ASP variables, default to false. (lb-founded Booleans)
- Concepts like trust, reputation, happiness etc. which need a reason to go up and by default might not exist. (lb-founded quantities)
- Investment in a company from a copycat investor. (lb-founded integer or real)
Some examples of founded quantities from modelling perspective are:

- ASP variables, default to \textit{false}. (lb-founded Booleans)
- Concepts like trust, reputation, happiness etc. which need a reason to go up and by default might not exist. (lb-founded quantities)
- Investment in a company from a copycat investor. (lb-founded integer or real)
- Variables representing shortest paths between nodes in a graph. (ub-founded integers or reals)
Argument against completion

\[ a \xrightarrow{10} b \rightleftharpoons c \]

It might appear that if we model the problem in CP with the following model, then we get the correct solutions:

\[
\begin{align*}
sp_a &= 0 \\
sp_b &= \min\{sp_a + 10, sp_c + 2\} \\
sp_c &= sp_b + 2
\end{align*}
\]

For example, \( sp_a = 0, sp_b = sp_c = 2 \) which is incorrect.
ub-lb conversion

Example

- A variable $a$ with a rule $a \leq k$ can be replaced by the variable $a' = -a$ and $a' \geq -k$.
- $sp_X$ can be replaced with $sp'_X = -sp_X$. The rule $sp_Y \leq sp_X + e_X, Y$ becomes $sp'_Y \geq e_X, Y - sp'_X$. 