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

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.

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.

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.

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.

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.

Motivation (Reachability)

Consider deciding which nodes are reachable from a: $a \qquad b \rightleftarrows c$

Motivation (Reachability)

Consider deciding which nodes are reachable from a: $a \quad b \rightleftarrows c$

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

$$
\begin{array}{rl}
 & r_a. \\
\forall (X, Y) \in E & r_Y \leftarrow r_X.
\end{array}
$$

Reachability (continued)

After grounding:

$$r_a.$$
$$r_b \leftarrow r_c.$$
$$r_c \leftarrow r_b.$$

Reachability (continued)

After grounding:

$$
\begin{aligned}
&r_a. \\
&r_b \leftarrow r_c. \\
&r_c \leftarrow r_b.
\end{aligned}
$$

A SAT solver would give the incorrect solution:
$r_a = r_b = r_c = \mathit{true}$.

Reachability (continued)

After grounding:

$$r_a.$$
$$r_b \leftarrow r_c.$$
$$r_c \leftarrow r_b.$$

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:

$$
\begin{aligned}
&r_a. \\
&r_b \leftarrow r_c. \\
&r_c \leftarrow r_b.
\end{aligned}
$$

A SAT solver would give the incorrect solution:
$r_a = r_b = r_c = \mathit{true}$.

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

They will detect $r_b = r_c = \mathit{true}$ 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:

$$\begin{tikzcd} a \arrow[r, "10"] & b \arrow[r, bend left, "2"] & c \arrow[l, bend left, "2"] \end{tikzcd} .$$

$$spub_a(0)$$

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:

$$\begin{tikzcd} a \arrow[r, "10"] & b \arrow[r, bend left, "2"] & c \arrow[l, bend left, "2"] \end{tikzcd} .$$

$$spub_a(0)$$

$$\forall X \qquad spub_X(M)$$

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:

$$\begin{tikzcd} a \arrow[r, "10"] & b \arrow[r, bend left, "2"] & c \arrow[l, bend left, "2"] \end{tikzcd} .$$

$$\begin{array}{ll} & spub_a(0) \\ \forall X & spub_X(M) \\ \forall (X, Y) \in E, S \in 0 \ldots M & \end{array}$$

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:

$$a \xrightarrow{10} b \underset{2}{\overset{2}{\rightleftarrows}} c\ .$$

$$\begin{array}{ll}
 & spub_a(0) \\
\forall X & spub_X(M) \\
\forall (X,Y) \in E, S \in 0 \ldots M & spub_Y(S+L) \leftarrow spub_X(S) + e_{X,Y}(L)
\end{array}$$

Shortest path (continued)

$$
\begin{tikzcd}
a \arrow[r, "10"] & b \arrow[r, bend left, "2"] & c \arrow[l, bend left, "2"]
\end{tikzcd}
$$

After grounding all nodes and edges:

$$
\begin{array}{ll}
 & spub_a(0) \\
 & spub_a(M), spub_b(M), spub_c(M) \\
\forall S \in 0..M & spub_b(10 + S) \leftarrow spub_a(S)
\end{array}
$$

Shortest path (continued)

$$\begin{tikzcd} a \arrow[r, "10"] & b \arrow[r, bend left, "2"] & c \arrow[l, bend left, "2"] \end{tikzcd}$$

After grounding all nodes and edges:

$$\begin{array}{ll}
 & spub_a(0) \\
 & spub_a(M), spub_b(M), spub_c(M) \\
\forall S \in 0..M & spub_b(10 + S) \leftarrow spub_a(S) \\
\forall S \in 0..M & spub_b(2 + S) \leftarrow spub_c(S)
\end{array}$$

Shortest path (continued)

$$\begin{tikzcd} a \arrow[r, "10"] & b \arrow[r, bend left, "2"] & c \arrow[l, bend left, "2"] \end{tikzcd}$$

After grounding all nodes and edges:

$$\begin{array}{ll}
 & spub_a(0) \\
 & spub_a(M), spub_b(M), spub_c(M) \\
\forall S \in 0..M & spub_b(10 + S) \leftarrow spub_a(S) \\
\forall S \in 0..M & spub_b(2 + S) \leftarrow spub_c(S) \\
\forall S \in 0..M & spub_c(2 + S) \leftarrow spub_b(S)
\end{array}$$

Shortest path (continued)

$$\begin{tikzcd} a \arrow[r, "10"] & b \arrow[r, bend left, "2"] & c \arrow[l, bend left, "2"] \end{tikzcd}$$

After grounding all nodes and edges:

$$\begin{array}{ll}
 & spub_a(0) \\
 & spub_a(M), spub_b(M), spub_c(M) \\
\forall S \in 0..M & spub_b(10 + S) \leftarrow spub_a(S) \\
\forall S \in 0..M & spub_b(2 + S) \leftarrow spub_c(S) \\
\forall S \in 0..M & spub_c(2 + S) \leftarrow spub_b(S)
\end{array}$$

A SAT solver allows solutions where $spub_a(0) = spub_b(0) = spub_c(0) = true$ 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.

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.

If there is no rule for a variable, then it must default to lowest or highest value.

Bound Founded Answer Set Programming

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.

Bound Founded Answer Set Programming

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 $y \in vars(c) \cap \mathcal{F}$.

Bound Founded Answer Set Programming

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 $y \in \mathit{vars}(c) \cap \mathcal{F}$.

Constraints can be written as rules with no heads $(c, -)$.

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 ∞ 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 ∞ 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

- *$a \leftarrow b \wedge \neg c$ 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

- $a \leftarrow b \wedge \neg c$ *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

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

HORN-CP

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.

HORN-CP

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.

Minimal assignments

Definition

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

Minimal assignments

Definition

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

Example

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

Minimal assignments

Definition

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

Example

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

Minimal assignments

Definition

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

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

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

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

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

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

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

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 θ, the CP-reduct of P w.r.t. θ (written P^θ) is a HORN-CP obtained by replacing in each rule $r = (c, y)$ every standard variable and every variable $v \in vars(c) \setminus \{y\}$ 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 θ, the CP-reduct of P w.r.t. θ (written P^θ) is a HORN-CP obtained by replacing in each rule $r = (c, y)$ every standard variable and every variable $v \in vars(c) \setminus \{y\}$ 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.*

$\theta = \{a \mapsto 7, c \mapsto 2, d \mapsto 10, b \mapsto 68\}$,
$P^\theta = \{a \geq 7, b \geq 10a - 2, c \geq 2\}$.

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

θ 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.*

$\theta = \{a \mapsto 7, c \mapsto 2, d \mapsto 10, b \mapsto 68\}$,
$P^\theta = \{a \geq 7, b \geq 10a - 2, c \geq 2\}$.

Stable assignments for BFASPs

Definition

θ 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.

$\theta = \{a \mapsto 7, c \mapsto 2, d \mapsto 10, b \mapsto 68\}$,
$P^\theta = \{a \geq 7, b \geq 10a - 2, c \geq 2\}$.

$MA(P^\theta) =_{\mathcal{F}} \theta$, therefore, θ is a stable assignment.

Shortest path in BFASP

$$\begin{tikzcd} a \arrow[r, "10"] & b \arrow[r, bend left, "2"] & c \arrow[l, bend left, "2"] \end{tikzcd}$$

Here is a ground BFASP model, sp_X is a ub-founded variable representing the shortest path from a to x:

$$sp_a \leq 0$$

Shortest path in BFASP

$$\begin{tikzcd} a \arrow[r, "10"] & b \arrow[r, bend left, "2"] & c \arrow[l, bend left, "2"] \end{tikzcd}$$

Here is a ground BFASP model, sp_X is a ub-founded variable representing the shortest path from a to x:

$$sp_a \leq 0$$
$$sp_b \leq sp_a + 10$$

Shortest path in BFASP

$$
\begin{tikzcd}
a \arrow[r, "10"] & b \arrow[r, bend left, "2"] & c \arrow[l, bend left, "2"]
\end{tikzcd}
$$

Here is a ground BFASP model, sp_X is a ub-founded variable representing the shortest path from a to x:

$$
\begin{aligned}
sp_a &\leq 0 \\
sp_b &\leq sp_a + 10 \\
sp_b &\leq sp_c + 2
\end{aligned}
$$

Shortest path in BFASP

$$\begin{tikzcd} a \arrow[r, "10"] & b \arrow[r, bend left, "2"] & c \arrow[l, bend left, "2"] \end{tikzcd}$$

Here is a ground BFASP model, sp_X is a ub-founded variable representing the shortest path from a to x:

$$\begin{aligned} &sp_a \leq 0 \\ &sp_b \leq sp_a + 10 \\ &sp_b \leq sp_c + 2 \\ &sp_c \leq sp_b + 2 \end{aligned}$$

Implementation

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 *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.

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.

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.

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.

Utilitarian policies (example)

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

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

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

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

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

Implementation

Consider the following utilitarian policies instance:

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

Implementation

Consider the following utilitarian policies instance:

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

Without stable model semantics, we can have solutions like $e_1 = e_2 = \mathit{false}, h_1 = 6, h2 = 7$, which do not make sense since no person was *proven* to be happy.

Implementation

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

Implementation

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

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

Implementation

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

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

Bound	Rule	Supports
$[h_1 \geq 5]$	$R1$	$[e_2 \neq \bot]$

Implementation

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

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

Bound	Rule	Supports
$[h_1 \geq 5]$	$R1$	$[e_2 \neq \bot]$
$[h_2 \geq 9]$	$R2$	$[e_1 \neq \bot]$

Implementation

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

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

Bound	Rule	Supports
$[h_1 \geq 5]$	$R1$	$[e_2 \neq \perp]$
$[h_2 \geq 9]$	$R2$	$[e_1 \neq \perp]$
$[h_1 \geq 11]$	$R1$	$[e_2 \neq \perp], [h_2 \geq 7]$

Implementation

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

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

Bound	Rule	Supports
$[h_1 \geq 5]$	$R1$	$[e_2 \neq \bot]$
$[h_2 \geq 9]$	$R2$	$[e_1 \neq \bot]$
$[h_1 \geq 11]$	$R1$	$[e_2 \neq \bot], [h_2 \geq 7]$
$[h_2 \geq 16]$	$R2$	$[e_1 \neq \bot], [h_1 \geq 5]$

Implementation

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

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

Bound	Rule	Supports
$[h_1 \geq 5]$	$R1$	$[e_2 \neq \perp]$
$[h_2 \geq 9]$	$R2$	$[e_1 \neq \perp]$
$[h_1 \geq 11]$	$R1$	$[e_2 \neq \perp], [h_2 \geq 7]$
$[h_2 \geq 16]$	$R2$	$[e_1 \neq \perp], [h_1 \geq 5]$

We propagate $[h_1 \leq 11]$ and $[h_2 \leq 16]$.

Implementation

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

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

Bound	Rule	Supports
$[h_1 \geq 5]$	$R1$	$[e_2 \neq \bot]$
$[h_2 \geq 9]$	$R2$	$[e_1 \neq \bot]$
$[h_1 \geq 11]$	$R1$	$[e_2 \neq \bot], [h_2 \geq 7]$
$[h_2 \geq 16]$	$R2$	$[e_1 \neq \bot], [h_1 \geq 5]$

Implementation (Dejustification)

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

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

Bound	Rule	Supports
$[h_1 \geq 5]$	$R1$	$[e_2 \neq \bot]$
$[h_1 \geq 11]$	$R1$	$[e_2 \neq \bot], [h_2 \geq 7]$
$[h_2 \geq 16]$	$R2$	$[e_1 \neq \bot], [h_1 \geq 5]$

Implementation (Dejustification)

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

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

Bound	Rule	Supports
$[h_1 \geq 5]$	$R1$	$[e_2 \neq \perp]$
$[h_1 \geq 11]$	$R1$	$[e_2 \neq \perp], [h_2 \geq 7]$

Implementation (Dejustification)

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

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

Bound	Rule	Supports
$[h_1 \geq 5]$	$R1$	$[e_2 \neq \perp]$

Implementation (Rejustification)

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

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

Bound	Rule	Supports
$[h_1 \geq 5]$	$R1$	$[e_2 \neq \perp]$
$[h_2 \geq 7]$	$R2$	$[h_1 \geq 5]$

Implementation (Rejustification)

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

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

Bound	Rule	Supports
$[h_1 \geq 5]$	$R1$	$[e_2 \neq \bot]$
$[h_2 \geq 7]$	$R2$	$[h_1 \geq 5]$
$[h_1 \geq 11]$	$R1$	$[e_2 \neq \bot], [h_2 \geq 7]$

Implementation (Rejustification)

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

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

Bound	Rule	Supports
$[h_1 \geq 5]$	$R1$	$[e_2 \neq \bot]$
$[h_2 \geq 7]$	$R2$	$[h_1 \geq 5]$
$[h_1 \geq 11]$	$R1$	$[e_2 \neq \bot], [h_2 \geq 7]$

We propagate $[h_2 \leq 7] \leftarrow [e_1 = \bot]$.

Implementation

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

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

Bound	Rule	Supports
$[h_1 \geq 5]$	$R1$	$[e_2 \neq \bot]$
$[h_2 \geq 7]$	$R2$	$[h_1 \geq 5]$
$[h_1 \geq 11]$	$R1$	$[e_2 \neq \bot], [h_2 \geq 7]$

Implementation (Dejustification)

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

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

Bound	Rule	Supports
$[h_2 \geq 7]$	$R2$	$[h_1 \geq 5]$
$[h_1 \geq 11]$	$R1$	$[e_2 \neq \perp], [h_2 \geq 7]$

Implementation (Dejustification)

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

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

Bound	Rule	Supports
$[h_2 \geq 7]$	$R2$	$[h_1 \geq 5]$

Implementation (Dejustification)

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

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

Implementation (Dejustification)

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

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

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

Experiments

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

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

Table : *Shortest path*

Experiments

			cl+gr				chuffed			
N	D	C	gr	opt	comp	time	flat	opt	comp	time
5	10	8	0.43	24	Y	0.65	0	24	Y	0.00
9	11	13	111.79	—	N	600	0.02	123	Y	0.06
25	45	40	—	—	—	—	0.66	2629	Y	3.46
45	25	32	—	—	—	—	4.91	2174531	N	600
60	30	25	—	—	—	—	13.15	6613781	N	600

Table : *Road construction*

Experiments

		cl+gr				chuffed			
S	C	gr	opt	comp	time	flat	opt	comp	time
10	5	0.00	102	Y	0.00	0.00	102	Y	0.00
50	10	0.05	702	N	600	0.02	234	Y	0.14
100	30	0.97	14701	N	600	0.22	408	Y	5.99
250	30	2.65	—	N	600	0.21	75	Y	35.52
500	50	16.63	—	N	600	0.72	—	N	600

Table : *Company controls*

Experiments

		cl+gr				chuffed			
C	P	gr	opt	comp	time	flat	opt	comp	time
3	5	.07	34	Y	.36	0.01	34	Y	0
5	6	66.64	—	N	600	0	680	Y	0
20	25	89.10	—	N	600	.06	886	Y	208.64
90	90	—	—	—	—	1.40	12628	N	600
250	230	—	—	—	—	13.74	—	N	600

Table : *Utilitarian policies*

Experiments

Scale	gr kB	cl+gr
1	748	0.36
2	2816	1.25
4	18741	37.83
6	67570	260.39
8	170287	600
10	336386	600

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.

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

Questions

Thank you!

Lower and upper bound founded variables

Some examples of founded quantities from modelling perspective are:

- ASP variables, default to *false*. (lb-founded Booleans)

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)

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)

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)
- Variables representing shortest paths between nodes in a graph. (ub-founded integers or reals)

Argument against completion

$$\begin{tikzcd} a \arrow[r, "10"] & b \arrow[r, bend left, "2"] & c \arrow[l, bend left, "1"] \end{tikzcd}$$

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

$$\begin{aligned} sp_a &= 0 \\ sp_b &= min\{sp_a + 10, sp_c + 2\} \\ sp_c &= sp_b + 2 \end{aligned}$$

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* $sp'_Y \geq e_{X,Y} - sp'_X$.