Satisfiability Modulo Theories
Lezione 5 - A Theory Solver for $\mathcal{IDL}$

(slides revision: Saturday 14^{th} March, 2015, 11:47)

Roberto Bruttomesso

Seminario di Logica Matematica
(Corso Prof. Silvio Ghilardi)

17 Novembre 2011

Recall from last lecture

The Lazy Approach for a theory $\mathcal{T}$ is based on the tight integration of

- a CDCL SAT-solver
- a $\mathcal{T}$-solver, a decision procedure for $\mathcal{T}$

Recall from last lecture

The Lazy Approach for a theory $\mathcal{T}$ is based on the tight integration of

- a CDCL SAT-solver
- a $\mathcal{T}$-solver, a decision procedure for $\mathcal{T}$

The SAT-solver enumerates Boolean models $\mu^{\mathcal{B}}$

The $\mathcal{T}$-solver checks the consistency of $\mu^{\mathcal{B}}$ in $\mathcal{T}$

Recall from last lecture

The Lazy Approach for a theory $\mathcal{T}$ is based on the tight integration of

- a CDCL SAT-solver
- a $\mathcal{T}$-solver, a decision procedure for $\mathcal{T}$

The SAT-solver enumerates Boolean models $\mu^{\mathcal{B}}$

The $\mathcal{T}$-solver checks the consistency of $\mu^{\mathcal{B}}$ in $\mathcal{T}$

For efficiency, it is desirable that the $\mathcal{T}$-solver

- Reasons incrementally (does not compute everything from scratch everytime)
- Backtracks efficiently
- Returns minimal conflicts
- Performs Theory-Propagation

Outline

1. Integer Difference Logic
   - Introduction
   - Translation into a Graph
   - Solving
2. Improvement for $\mathcal{T}$-solver
   - Minimal Conflicts
   - Incrementality
   - Backtrackability
   - Theory Propagation
3. Final Remarks

Integer Difference Logics ($\mathcal{IDL}$)

The $\mathcal{T}$-atoms of $\mathcal{IDL}$ consists of arithmetic constraints of the form

$$x - y \leq c$$

where x, y are variables and c is a numerical constant. The domain of x, y, c is that of the integers

Integer Difference Logics ($\mathcal{IDL}$)

The $\mathcal{T}$-atoms of $\mathcal{IDL}$ consists of arithmetic constraints of the form

$$x - y \leq c$$

where x, y are variables and c is a numerical constant. The domain of x, y, c is that of the integers

Notice that the following translations hold

- $x - y \geq c \quad \Longrightarrow \quad y - x \leq -c$
- $x - y < c \quad \Longrightarrow \quad x - y \leq c - 1$
- $x - y > c \quad \Longrightarrow \quad y - x \leq -c - 1$
- $x - y = c \quad \Longrightarrow \quad (x - y \leq c) \wedge (x - y \geq c)$
- $x - y \neq c \quad \Longrightarrow \quad (x - y < c) \vee (x - y > c)$

Integer Difference Logic ($\mathcal{IDL}$)

$\mathcal{RDL}$ is similar to $\mathcal{IDL}$, but it is defined on the rationals. However an $\mathcal{RDL}$ formula can be reduced to an equisatisfiable $\mathcal{IDL}$ formula

Integer Difference Logic ($\mathcal{IDL}$)

$\mathcal{RDL}$ is similar to $\mathcal{IDL}$, but it is defined on the rationals. However an $\mathcal{RDL}$ formula can be reduced to an equisatisfiable $\mathcal{IDL}$ formula

$\mathcal{IDL}/\mathcal{RDL}$ can be used to encode a large variety of verification probems

- scheduling
- TSP
- ASP
- timed-automata
- sorting algorithms

Also, the worst-case complexity of solving a conjunction of $\mathcal{IDL}$ constraints is $O(m + n \log n)$

Translation into a Graph

The constraint $x - y \leq c$ says that

"the distance between x and y is at most c"

This can be encoded as $(x, y; c)$

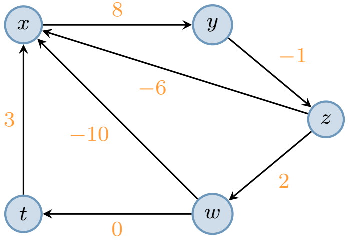
The constraint $x - y \leq c$ says that

"the distance between x and y is at most c"

This can be encoded as $(x, y; c)$

So, a set $\mu^{\mathcal{B}}$ can be encoded as a graph. Concrete example:

$$\begin{array}{rcr} x - y & \leq & 8 \\ y - z & \leq & -1 \\ z - x & \leq & -6 \\ z - w & \leq & 2 \\ w - x & \leq & -10 \\ w - t & \leq & 0 \\ t - x & \leq & 3 \end{array}$$



$G(V, E)$: $V = \{x, y, z, w, t\}$, $E = \{(x, y; 8), (y, z; -1), (z, x; -6), (z, w; 2), (w, x; -10), \ldots\}$

Translation into a Graph

Theorem (Translation)

$\mu^{\mathcal{B}}$ *is* $\mathcal{IDL}$*-unsatisfiable*

iff

there is a **negative cycle** *in the corresponging graph* $G(V, E)$

Translation into a Graph

Theorem (Translation)

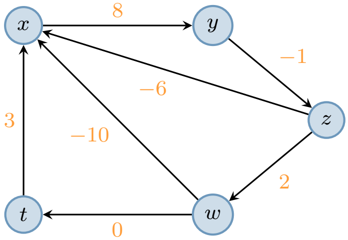
$\mu^{\mathcal{B}}$ is $\mathcal{IDL}$-unsatisfiable

iff

there is a **negative cycle** *in the corresponging graph $G(V, E)$*

E.g.:

$$\begin{array}{rcr} x - y & \leq & 8 \\ y - z & \leq & -1 \\ z - x & \leq & -6 \\ z - w & \leq & 2 \\ w - x & \leq & -10 \\ w - t & \leq & 0 \\ t - x & \leq & 3 \end{array}$$



Translation into a Graph

Theorem (Translation)

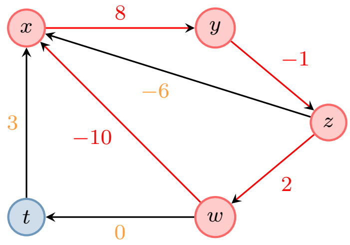
$\mu^{\mathcal{B}}$ is $\mathcal{IDL}$-unsatisfiable

iff

there is a **negative cycle** *in the corresponging graph $G(V, E)$*

E.g.:

$$\begin{array}{rcr} x - y & \leq & 8 \\ y - z & \leq & -1 \\ z - x & \leq & -6 \\ z - w & \leq & 2 \\ w - x & \leq & -10 \\ w - t & \leq & 0 \\ t - x & \leq & 3 \end{array}$$



Translation into a Graph

We use the following lemma to aid the proof

Lemma (Farka's Lemma for $\mathcal{IDL}$)

$\mu^{\mathcal{B}}$ is unsatisfiable iff there exists a subset
$\nu^{\mathcal{B}} = \{\ x_1 - x_2 \leq c_1,\ x_2 - x_3 \leq c_2,\ \ldots,\ x_n - x_1 \leq c_n\ \}$ of $\mu^{\mathcal{B}}$ such that
$c_1 + \ldots + c_n < 0$

Translation into a Graph

We use the following lemma to aid the proof

Lemma (Farka's Lemma for $\mathcal{IDL}$)

$\mu^{\mathcal{B}}$ is unsatisfiable iff there exists a subset $\nu^{\mathcal{B}} = \{\ x_1 - x_2 \leq c_1,\ x_2 - x_3 \leq c_2,\ \ldots,\ x_n - x_1 \leq c_n\ \}$ of $\mu^{\mathcal{B}}$ such that $c_1 + \ldots + c_n < 0$

The proof of Theorem is now trivial

Proof.

$\mu^{\mathcal{B}}$ is unsatisfiable iff by Farka's Lemma for $\mathcal{IDL}$
there exists $\nu^{\mathcal{B}} \subseteq \mu^{\mathcal{B}}$ with $c_1 + \ldots + c_n < 0$ iff by our translation
there exists a negative cycle in $G(V, E)$ $\square$

Suppose that no negative cycle exists in $G(V,E)$ for $\mu^{\mathcal{B}}$, how do we find a model μ for the integer variables ?

Solving

Suppose that no negative cycle exists in $G(V, E)$ for $\mu^{\mathcal{B}}$, how do we find a model μ for the integer variables ?

State-of-the-art methods employ SSSP (Single Source Shortest Paths) algorithms, such as the **Bellman-Ford** algorithm (or its variations), as follows:

- we add an artificial vertex I to V, and add $\{(I, x_1; 0), \ldots, (I, x_n; 0)\}$ to E (notice that this never creates negative cycles)
- Compute the shortest paths from source I (run Bellman-Ford). Let $\pi(x)$ be the shortest path from I to x, for all $x \in V$
- The model μ can be computed as $\mu(x) = -\pi(x)$ for all $x \in V$ (see later why)

Bellman-Ford

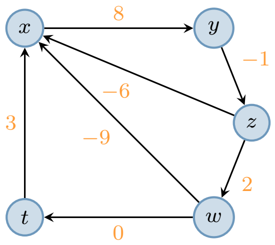
$\pi(x)$: current distance of x from I
TBV: queue of vertexes To Be Visited

$\pi(x) = \infty$ for all $x \in V, x \neq I$
$\pi(I) = 0$
$TBV.pushBack(I)$
while ($TBV.size(\ ) > 0$)
    $s = TBV.popFront(\ )$
    foreach $(s, t; w) \in E$ // for each outgoing edge
        if ($\pi(t) - \pi(s) > w$) // is too far ?
            $\pi(t) = \pi(s) + w$ // relax (decrease $\pi(t)$)
            if ($TBV.has(t) == false$)
                $TBV.pushBack(t)$ // enqueue t if not there

Bellman-Ford, Example

$\pi(x)$: current distance of x from I
TBV: queue of vertexes To Be Visited

$\pi(x) = \infty$ for all $x \in V, x \neq I$
$\pi(I) = 0$
$TBV.pushBack(I)$
while ($TBV.size(\ ) > 0$)
  $s = TBV.popFront(\ )$
  foreach $(s, t; w) \in E$
    if ($\pi(t) - \pi(s) > w$)
      $\pi(t) = \pi(s) + w$
      if ($TBV.has(t) == false$)
        $TBV.pushBack(t)$

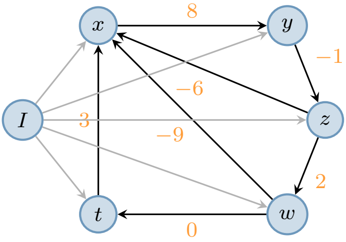


$TBV = [\ ]$
Current vertex:
Current edge:

Bellman-Ford, Example

$\pi(x)$: current distance of x from I
TBV: queue of vertexes To Be Visited

$\pi(x) = \infty$ for all $x \in V, x \neq I$
$\pi(I) = 0$
$TBV.pushBack(I)$
while ($TBV.size(\ ) > 0$)
  $s = TBV.popFront(\ )$
  foreach $(s, t; w) \in E$
    if ($\pi(t) - \pi(s) > w$)
      $\pi(t) = \pi(s) + w$
      if ($TBV.has(t) == false$)
        $TBV.pushBack(t)$

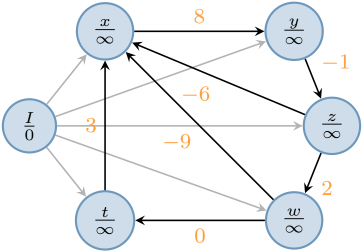


$TBV = [\ \ ]$
Current vertex:
Current edge:

Bellman-Ford, Example

$\pi(x)$: current distance of x from I
TBV: queue of vertexes To Be Visited

$\pi(x) = \infty$ for all $x \in V, x \neq I$
$\pi(I) = 0$
$TBV.pushBack(I)$
while ($TBV.size(\ ) > 0$)
  $s = TBV.popFront(\ )$
  foreach $(s, t; w) \in E$
    if ($\pi(t) - \pi(s) > w$)
      $\pi(t) = \pi(s) + w$
      if ($TBV.has(t) == false$)
        $TBV.pushBack(t)$

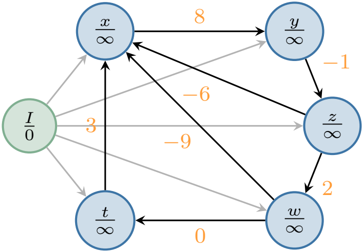


$TBV = [\ I\ ]$
Current vertex:
Current edge:

Bellman-Ford, Example

$\pi(x)$: current distance of x from I
TBV: queue of vertexes To Be Visited

$\pi(x) = \infty$ for all $x \in V, x \neq I$
$\pi(I) = 0$
$TBV.pushBack(I)$
while ($TBV.size(\ ) > 0$)
  $s = TBV.popFront(\ )$
  foreach $(s, t; w) \in E$
    if ($\pi(t) - \pi(s) > w$)
      $\pi(t) = \pi(s) + w$
      if ($TBV.has(t) == false$)
        $TBV.pushBack(t)$

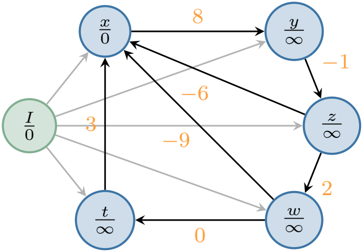


$TBV = [\ ]$
Current vertex: I
Current edge:

Bellman-Ford, Example

$\pi(x)$: current distance of x from I
TBV: queue of vertexes To Be Visited

$\pi(x) = \infty$ for all $x \in V, x \neq I$
$\pi(I) = 0$
$TBV.pushBack(I)$
while ($TBV.size(\,) > 0$)
  $s = TBV.popFront(\,)$
  foreach $(s, t; w) \in E$
    if ($\pi(t) - \pi(s) > w$)
      $\pi(t) = \pi(s) + w$
      if ($TBV.has(t) == false$)
        $TBV.pushBack(t)$

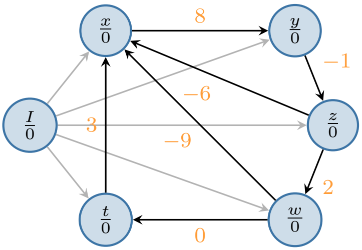


$TBV = [\ x\ ]$
Current vertex: I
Current edge: $(I, x; 0)$

Bellman-Ford, Example

$\pi(x)$: current distance of x from I
TBV: queue of vertexes To Be Visited

$\pi(x) = \infty$ for all $x \in V, x \neq I$
$\pi(I) = 0$
$TBV.pushBack(I)$
while ($TBV.size(\ ) > 0$)
  $s = TBV.popFront(\ )$
  foreach $(s, t; w) \in E$
    if ($\pi(t) - \pi(s) > w$)
      $\pi(t) = \pi(s) + w$
      if ($TBV.has(t) == false$)
        $TBV.pushBack(t)$

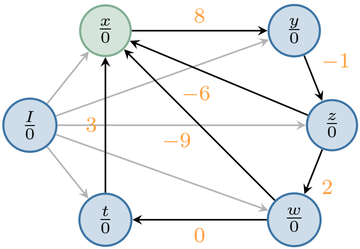


$TBV = [\ x, y, z, w, t\ ]$
Current vertex:
Current edge:

Bellman-Ford, Example

$\pi(x)$: current distance of x from I
TBV: queue of vertexes To Be Visited

$\pi(x) = \infty$ for all $x \in V, x \neq I$
$\pi(I) = 0$
$TBV.pushBack(I)$
while ($TBV.size(\ ) > 0$)
  $s = TBV.popFront(\ )$
  foreach $(s, t; w) \in E$
    if ($\pi(t) - \pi(s) > w$)
      $\pi(t) = \pi(s) + w$
      if ($TBV.has(t) == false$)
        $TBV.pushBack(t)$

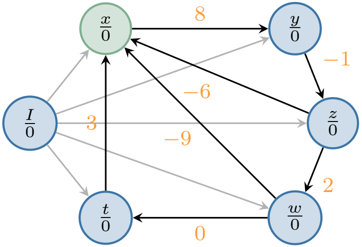


$TBV = [\ y, z, w, t\ ]$
Current vertex: x
Current edge:

Bellman-Ford, Example

$\pi(x)$: current distance of x from I
TBV: queue of vertexes To Be Visited

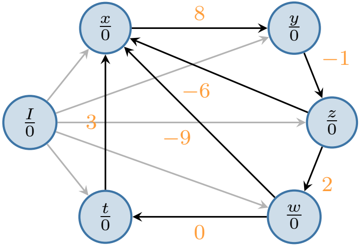
$\pi(x) = \infty$ for all $x \in V, x \neq I$
$\pi(I) = 0$
$TBV.pushBack(I)$
while ($TBV.size(\ ) > 0$)
  $s = TBV.popFront(\ )$
  foreach $(s, t; w) \in E$
    if ($\pi(t) - \pi(s) > w$)
      $\pi(t) = \pi(s) + w$
      if ($TBV.has(t) == false$)
        $TBV.pushBack(t)$



$TBV = [\ y, z, w, t\ ]$
Current vertex: x
Current edge:

$\pi(x)$: current distance of x from I
TBV: queue of vertexes To Be Visited

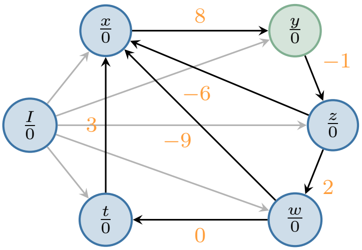
$\pi(x) = \infty$ for all $x \in V, x \neq I$
$\pi(I) = 0$
$TBV.pushBack(I)$
while ($TBV.size(\ ) > 0$)
  $s = TBV.popFront(\ )$
  foreach $(s, t; w) \in E$
    if ($\pi(t) - \pi(s) > w$)
      $\pi(t) = \pi(s) + w$
      if ($TBV.has(t) == false$)
        $TBV.pushBack(t)$



$TBV = [\ y, z, w, t\ ]$
Current vertex:
Current edge:

$\pi(x)$: current distance of x from I
TBV: queue of vertexes To Be Visited

$\pi(x) = \infty$ for all $x \in V, x \neq I$
$\pi(I) = 0$
$TBV.pushBack(I)$
while ($TBV.size(\,) > 0$)
  $s = TBV.popFront(\,)$
  foreach $(s, t; w) \in E$
    if ($\pi(t) - \pi(s) > w$)
      $\pi(t) = \pi(s) + w$
      if ($TBV.has(t) == false$)
        $TBV.pushBack(t)$

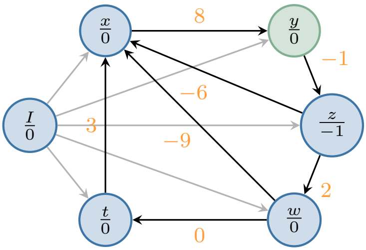


$TBV = [\ z, w, t\ ]$
Current vertex: y
Current edge:

Bellman-Ford, Example

$\pi(x)$: current distance of x from I
TBV: queue of vertexes To Be Visited

$\pi(x) = \infty$ for all $x \in V, x \neq I$
$\pi(I) = 0$
$TBV.pushBack(I)$
while ($TBV.size(\ ) > 0$)
  $s = TBV.popFront(\ )$
  foreach $(s, t; w) \in E$
    if ($\pi(t) - \pi(s) > w$)
      $\pi(t) = \pi(s) + w$
      if ($TBV.has(t) == false$)
        $TBV.pushBack(t)$

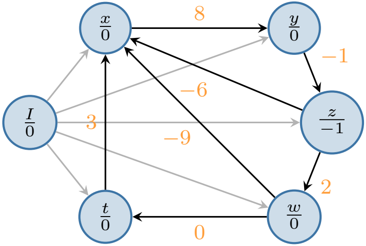


$TBV = [\ z, w, t\ ]$
Current vertex: y
Current edge: $(y, z; -1)$

Bellman-Ford, Example

$\pi(x)$: current distance of x from I
TBV: queue of vertexes To Be Visited

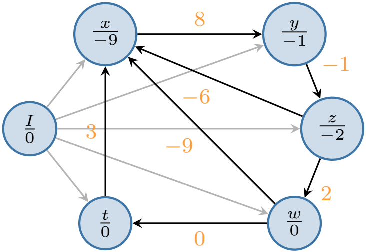
$\pi(x) = \infty$ for all $x \in V, x \neq I$
$\pi(I) = 0$
$TBV.pushBack(I)$
while ($TBV.size(\,) > 0$)
  $s = TBV.popFront(\,)$
  foreach $(s, t; w) \in E$
    if ($\pi(t) - \pi(s) > w$)
      $\pi(t) = \pi(s) + w$
      if ($TBV.has(t) == false$)
        $TBV.pushBack(t)$



$TBV = [\ z, w, t\ ]$
Current vertex:
Current edge:

$\pi(x)$: current distance of x from I
TBV: queue of vertexes To Be Visited

$$
\begin{array}{l}
\pi(x) = \infty \text{ for all } x \in V, x \neq I \\
\pi(I) = 0 \\
TBV.pushBack(I) \\
\text{while } (\ TBV.size(\ ) > 0\ ) \\
\quad s = TBV.popFront(\ ) \\
\quad \text{foreach } (s, t; w) \in E \\
\quad\quad \text{if } (\ \pi(t) - \pi(s) > w\ ) \\
\quad\quad\quad \pi(t) = \pi(s) + w \\
\quad\quad\quad \text{if } (\ TBV.has(t) == false\ ) \\
\quad\quad\quad\quad TBV.pushBack(t)
\end{array}
$$



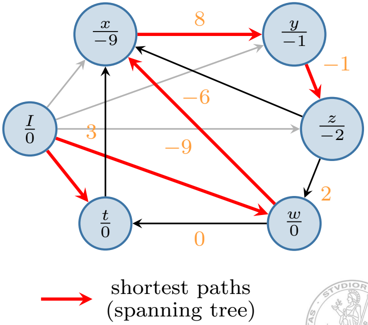
$TBV = [\ \ ]$
Current vertex:
Current edge:

Bellman-Ford, Example

$\pi(x)$: current distance of x from I
TBV: queue of vertexes To Be Visited

$\pi(x) = \infty$ for all $x \in V, x \neq I$
$\pi(I) = 0$
$TBV.pushBack(I)$
while ($TBV.size(\ ) > 0$)
  $s = TBV.popFront(\ )$
  foreach $(s, t; w) \in E$
    if ($\pi(t) - \pi(s) > w$)
      $\pi(t) = \pi(s) + w$
      if ($TBV.has(t) == false$)
        $TBV.pushBack(t)$

$TBV = [\ \ ]$
Current vertex:
Current edge:



Bellman-Ford, consideration

Invariant

At the end of BF, $\pi(x)$ holds the shortest distance from I to x, for all $x \in V$

Bellman-Ford, consideration

Invariant

At the end of BF, $\pi(x)$ holds the shortest distance from I to x, for all $x \in V$

Lemma (Shortest Path)

At the end of BF, $\pi(y) - \pi(x) \leq c$ *holds for all* $(x, y; c) \in E$

Bellman-Ford, consideration

Invariant

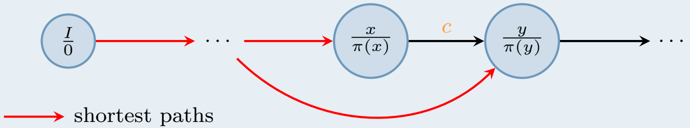
At the end of BF, $\pi(x)$ holds the shortest distance from I to x, for all $x \in V$

Lemma (Shortest Path)

At the end of BF, $\pi(y) - \pi(x) \leq c$ *holds for all* $(x, y; c) \in E$

Proof.

Suppose, for the sake of contradiction, that for an edge $(x, y; c)$, we have $\pi(y) - \pi(x) > c$



$\pi(x)$ is the shortest dist. from I to x (by Invariant). But since $\pi(y) > \pi(x) + c$, the shortest path from I to y is $\pi(x) + c$. So $\pi(y)$ is not the shortest dist. Contradiction. $\square$

Finding a model μ

Because of the previous lemma we have that

$$\pi(y) - \pi(x) \leq c \qquad \text{holds for all } (x, y; c) \in E$$

So, if we take $\qquad \mu(x) = -\pi(x) \qquad$ we have that

$$\mu(x) - \mu(y) \leq c \qquad \text{holds for all constraints in } \mu^{\mathcal{B}}$$

and therefore μ is a model

Outline

$\mathcal{T}$-solver Compliance

So far we have seen that BF can be used to compute a model of a given **consistent** set of $\mathcal{IDL}$ constraints

Recall that to have an efficient $\mathcal{T}$-solver the following features should be supported

- Minimal Conflicts
- Incrementality
- Backtrackability
- Theory Propagation

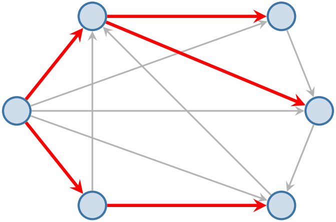
Let's see how to improve the current algorithm to support them

Minimal Conflicts

So far we assumed that $G(V, E)$ did not contain negative cycles. However it is not difficult to tweak the BF to recognize them

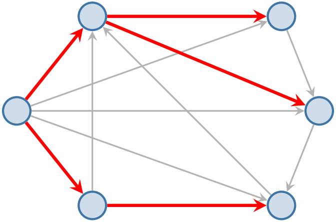
Minimal Conflicts

So far we assumed that $G(V, E)$ did not contain negative cycles.
However it is not difficult to tweak the BF to recognize them
Recall the **spanning tree**: the tree that holds shortest paths



Minimal Conflicts

So far we assumed that $G(V, E)$ did not contain negative cycles.
However it is not difficult to tweak the BF to recognize them
Recall the **spanning tree**: the tree that holds shortest paths



It is easy to keep the spanning tree:

- each node t keeps two fields **fatherVertex** and **fatherEdge** that stores its father in the spanning tree
- when a **relax** is done on t (line 8 of BF) through an edge $e = (s, t; c)$, $t.fatherVertex$ is set to s and $t.fatherEdge$ is set to e

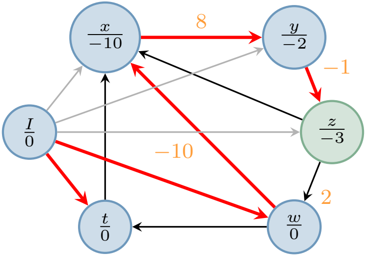
Minimal Conflicts

When a negative cycle exists, the spanning tree tends to become **cyclic** (trees are acyclic instead). It is easy to recognize this situation and report the negative cycle

Minimal Conflicts

When a negative cycle exists, the spanning tree tends to become **cyclic** (trees are acyclic instead). It is easy to recognize this situation and report the negative cycle

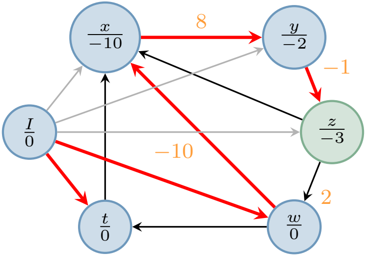
$TBV = [\,]$
Current vetex: z
Current edge: $(z, w; 2)$



Minimal Conflicts

When a negative cycle exists, the spanning tree tends to become **cyclic** (trees are acyclic instead). It is easy to recognize this situation and report the negative cycle

$TBV = [\,]$
Current vetex: z
Current edge: $(z, w; 2)$



checkNegativeCycle(s, t)
$\nu^{\mathcal{B}} = \{\ \}$

while ($s \neq t$ && $s \neq I$)
$s = s.fatherVertex$
$\nu^{\mathcal{B}} = \nu^{\mathcal{B}} \cup \{s.fatherEdge\}$

if ($s == t$) return conflict // Neg. Cycle detected
return OK // I reached

Minimal Conflicts

Bellman-Ford with negative cycle detection

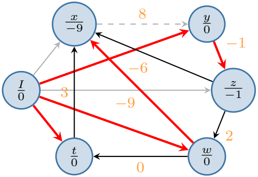
```
π(x) = ∞ for all x ∈ V, x ≠ I
π(I) = 0
TBV.pushBack(I)
while ( TBV.size( ) > 0 )
  s = TBV.popFront( )
  foreach (s, t; w) ∈ E                          // for each outgoing edge
    if ( π(t) − π(s) > w )                       // is too far ?
      if ( checkNegativeCycle( s, t ) == conflict )
        return ν^B
      s.fatherVertex = t
      s.fatherEdge = (s, t; w)
      π(t) = π(s) + w                            // relax (decrease π(t))
      if ( TBV.has(t) == false )
        TBV.pushBack(t)                          // enqueue t if not there
```

Incrementality

Incrementality comes almost for free !

- we always keep the π function "alive", and only update it slightly
- $G(V, E)$ keeps track of **active** and **inactive** edges: active edges are those on $\mu^{\mathcal{B}}$
- when a new $\mathcal{T}$-atom $x - y \leq c$ is added to $\mu^{\mathcal{B}}$, the corresponding edge $(x, y; c)$ becomes active in the graph
- we add x to TBV, and run BF from line 4

$x - y \leq 8 \notin \mu^{\mathcal{B}}$

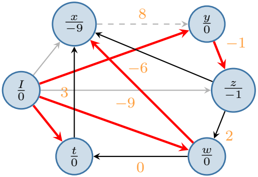


Incrementality

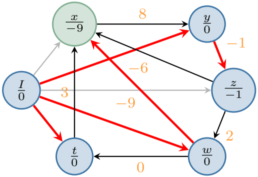
Incrementality comes almost for free !

- we always keep the π function "alive", and only update it slightly
- $G(V, E)$ keeps track of **active** and **inactive** edges: active edges are those on $\mu^{\mathcal{B}}$
- when a new $\mathcal{T}$-atom $x - y \leq c$ is added to $\mu^{\mathcal{B}}$, the corresponding edge $(x, y; c)$ becomes active in the graph
- we add x to TBV, and run BF from line 4

$x - y \leq 8 \notin \mu^{\mathcal{B}}$



$x - y \leq 8$ pushed to $\mu^{\mathcal{B}}$

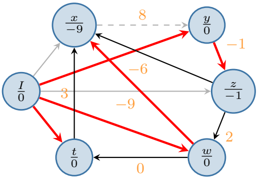


Incrementality

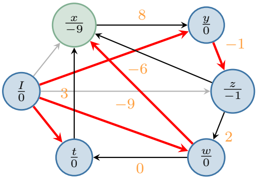
Incrementality comes almost for free !

- we always keep the π function "alive", and only update it slightly
- $G(V, E)$ keeps track of **active** and **inactive** edges: active edges are those on $\mu^{\mathcal{B}}$
- when a new $\mathcal{T}$-atom $x - y \leq c$ is added to $\mu^{\mathcal{B}}$, the corresponding edge $(x, y; c)$ becomes active in the graph
- we add x to TBV, and run BF from line 4

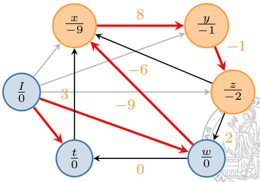
$x - y \leq 8 \notin \mu^{\mathcal{B}}$



$x - y \leq 8$ pushed to $\mu^{\mathcal{B}}$



After BF



Backtrackability

Backtracking can be done efficiently

First of all, recall that $\mu(x) = -\pi(x)$. Second obeserve that

Observation 1

Let μ be a model for a set $\mu^{\mathcal{B}}$ of constraints. Then μ is also a model for **any subset** of $\mu^{\mathcal{B}}$

Observation 1 implies that when backtracking we just have to turn some edges into inactive, and keep the last π intact

This is done as follows: BF will always work on a temporary π, called π'. If satisfiable, π is replaced by π', otherwise we keep π

Theory Propagation

Theory Propagation is the process of activating some edges in the graph that are implied by the current $\mu^{\mathcal{B}}$

Observation 2

A set of constraints

$\{ x_1 - x_2 \leq c_1,\ x_2 - x_3 \leq c_2,\ \ldots,\ x_{n-1} - x_n \leq c_{n-1} \}$

implies $\quad x_1 - x_n \leq c_n \quad$ iff $\quad c_1 + c_2 + \ldots c_{n-1} \leq c_n$

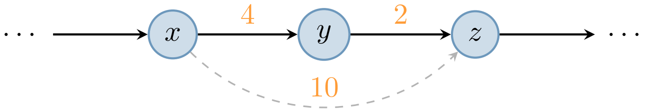
Theory Propagation

Theory Propagation is the process of activating some edges in the graph that are implied by the current $\mu^{\mathcal{B}}$

Observation 2

A set of constraints
$\{\ x_1 - x_2 \leq c_1,\ x_2 - x_3 \leq c_2,\ \ldots,\ x_{n-1} - x_n \leq c_{n-1}\ \}$
implies $\quad x_1 - x_n \leq c_n \quad$ iff $\quad c_1 + c_2 + \ldots c_{n-1} \leq c_n$

Example:



$x - z \leq 10$, currently inactive, can be theory-propagated

Final Remarks

We did not say how to handle negative $\mathcal{T}$-atoms, such as $\neg(x - y \leq c)$. However it is easy to see that $\quad \neg(x - y \leq c) \quad$ iff $\quad y - x \leq -c - 1$

Each $\mathcal{T}$-atom is associated to two edges $(x, y; c), (y, x; -c - 1)$. However at most only one of the two is activated (depending if $\mathcal{T}$-atom is pushed positively or negatively)

Final Remarks

We did not say how to handle negative $\mathcal{T}$-atoms, such as $\neg(x - y \le c)$. However it is easy to see that $\quad \neg(x - y \le c) \quad$ iff $\quad y - x \le -c - 1$

Each $\mathcal{T}$-atom is associated to two edges $(x, y; c), (y, x; -c - 1)$. However at most only one of the two is activated (depending if $\mathcal{T}$-atom is pushed positively or negatively)

Simple bounds, such as $\quad x \le c \quad$ or $\quad -x \le c \quad$ can also be handled. It is sufficient to add a "fake" (and fresh) variable Z (stands for "zero"), and use $\quad x - Z \le c \quad$ and $\quad Z - x \le c \quad$ instead of the above

Floyd-Warshall

Another algorithm that can be used instead of BF is the Floyd-Warshall

FW has a complexity of $\theta(n^3)$, while BF of $O(nm)$ (variations of BF have complexity $O(m + n \log n)$)

FW however is useful as it is trivial to compute all theory propagations. In BF, theory propagations are tricky to discover

FW is independent of m, the numeber of edges. FW is good for **dense** problems, while is likely to be bad for **sparse** ones

Exercizes

1. Prove that adding $\{(I, x_1; 0), \ldots, (I, x_n; 0)\}$ to a graph free of negative cycles, does not introduce any negative cycle
2. Let $G(V, E)$ be a graph, let $x, y \in V$, and suppose that a path $y \to \ldots \to x$ exists in G. Are the following true or false ? (if false, show counterexamples)
   (a) Adding an edge $(x, y; 100)$ to G never causes the creation of a negative cycle
   (b) Adding an edge $(x, y; -100)$ to G always causes the creation of a negative cycle
3. Let $\mu^{\mathcal{B}}$ be a set of constraints of the form $x - y \leq c$. Let μ be a model. Let $\zeta(x) = \{\mu(x) + \epsilon\}$, for some $\epsilon \in \mathbb{Z}$. Show that ζ is also a model
4. Show that conflicts returned discovered with checkNegativeCycle are always minimal
5. Prove Observation 2 using Farka's Lemma