An Incremental Approach to Structured Argumentation over Dynamic Knowledge Bases

Gianvincenzo Alfano[1], Sergio Greco[1], Francesco Parisi[1], Gerardo Ignacio Simari[2], Guillermo Ricardo Simari[2]

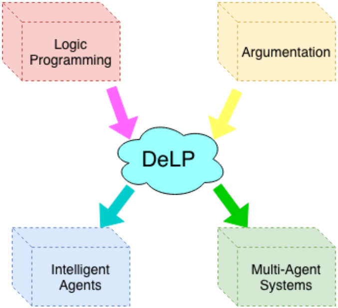
[1] Department of Informatics, Modeling, Electronics and System Engineering
University of Calabria, Italy
{g.alfano, greco, fparisi}@dimes.unical.it
[2] Departamento de Ciencias e Ing. de la Computación
Universidad Nacional del Sur (UNS)
Instituto de Ciencias e Ing. de la Computación (ICIC UNS−CONICET) Argentina
{gis,grs}@cs.uns.edu.ar

16*th* International Conference on Principles of Knowledge Representation and Reasoning

October 30-November 2, 2018
Tempe, Arizona

Dynamic Structured Argumentation

- Argumentation frameworks are often dynamic (change over time) as a consequence of the fact that argumentation is inherently dynamic (change mind/opinion, new available knowledge)
- We focus on **De**feasible **L**ogic **P**rogramming, a formalism that combines results of Logic Programming and Defeasible Argumentation.
- We devise an incremental technique for computing conclusions in structured argumentation frameworks (avoiding wasted effort due to recomputation from scratch)



Outline

A DeLP Program

DeLP considers two kinds of program rules:

- **Defeasible** rules to represent tentative information, and
- **Strict** rules used to represent strict knowledge.

Example (A DeLP-program $\mathcal{P}_1$)

Consider the DeLP-program $\mathcal{P}_1 = (\Pi_1, \Delta_1)$, where:

$$\Pi_1 = \{\sim a,\ t,\ b,\ (d \leftarrow t)\}$$

$$\Delta_1 = \left\{ \begin{array}{lll} (i \prec s), & (s \prec h), & (h \prec b), \\ (\sim h \prec d, t), & (\sim i \prec \sim a, s), & (a \prec t), \\ (s \prec d), & (h \prec d), & (\sim f \prec \sim e), \\ (\sim e \prec \sim h, \sim a) & & \end{array} \right\}$$

Argument

Given a DeLP program $\mathcal{P} = (\Pi, \Delta)$ and a literal α, we say that $\langle \mathcal{A}, \alpha \rangle$ is an argument for α if $\mathcal{A}$ is a set of defeasible rules of Δ such that:

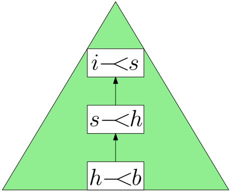
(i) there is a derivation for α from $\Pi \cup \mathcal{A}$,

(ii) the set $\Pi \cup \mathcal{A}$ is not contradictory, and

(iii) $\mathcal{A}$ is minimal (i.e., there is no proper subset $\mathcal{A}'$ of $\mathcal{A}$ satisfying both (i) and (ii)).

Example (An argument for $\mathcal{P}_1$)

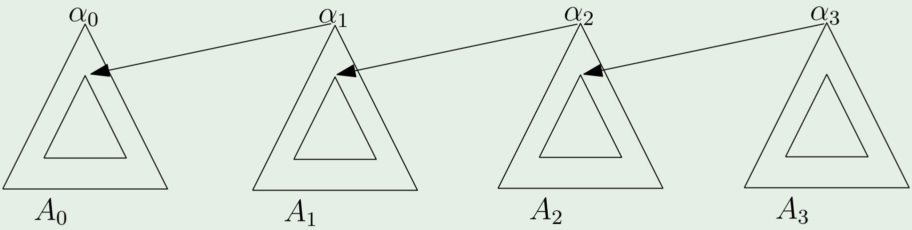
- $\langle \mathcal{A}_1, i \rangle = \langle \{(i \mathrel{-\!\!\prec} s),\ (s \mathrel{-\!\!\prec} h),\ (h \mathrel{-\!\!\prec} b)\}, i \rangle$



Argumentation Line

- A sequence of arguments obtained from a DeLP program, where each element of the sequence is a **defeater** of its predecessor.

Example (Argumentation Line)



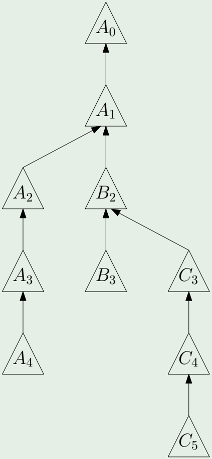
Example (An argumentation line for $\mathcal{P}_1$)

Given the two arguments: $\langle \mathcal{A}_1, i \rangle = \langle \{(i \prec s),\ (s \prec h),\ (h \prec b)\}, i \rangle$, and
$\langle \mathcal{A}_2, \sim i \rangle = \langle \{(\sim i \prec \sim a, s),\ (s \prec d)\}, \sim i \rangle$
an argumentation line is the following: $[\mathcal{A}_1, \mathcal{A}_2]$

Dialectical Process

- Given an argument A for a literal L, the dialectical tree contains all acceptable argumentation lines that start with that argument.
- It allows to determine the status for a given argument.

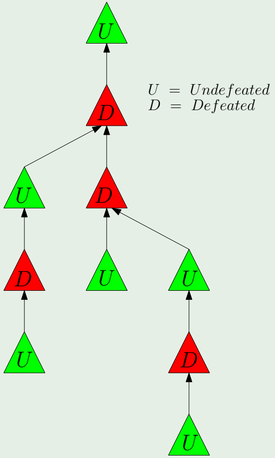
Example (Dialectical Tree)



Dialectical Process

- All leaves are marked as **Undefeated**.
- An argument in the tree is marked as **Defeated** if and only if it has at least a child marked as **Undefeated**.

Example (Dialectical Tree)



Status of literals

The status of literals allow us to determine the conclusions we can draw from a DeLP-program.

$S_{\mathcal{P}} : Lit \rightarrow \{\text{IN}, \text{OUT}, \text{UNDECIDED}, \text{UNKNOWN}\}$ assigning a *status* to each literal w.r.t. $\mathcal{P}$ as follows:

- $S_{\mathcal{P}}(\alpha) = \text{IN}$ if there exists a (marked) dialectical tree whose root α is *Undefeated*
- $S_{\mathcal{P}}(\alpha) = \text{OUT}$ if $S_{\mathcal{P}}(\sim\alpha) = \text{IN}$
- $S_{\mathcal{P}}(\alpha) = \text{UNDECIDED}$ if neither $S_{\mathcal{P}}(\alpha) = \text{IN}$ nor $S_{\mathcal{P}}(\alpha) = \text{OUT}$
- $S_{\mathcal{P}}(\alpha) = \text{UNKNOWN}$ if $\alpha \notin Lit_{\mathcal{P}}$, i.e., α is not in the language of the program

Example (Arguments from the previous program)

Given $\mathcal{P}$, then $S_{\mathcal{P}_1}(h) = \text{IN}$, $S_{\mathcal{P}_1}(a) = \text{OUT}$, and $S_{\mathcal{P}_1}(i) = \text{UNDECIDED}$.

Outline

Theorem Given $\mathcal{P}$ and a literal $\alpha \in Lit_{\mathcal{P}}$, deciding whether there is an argument for α w.r.t. $\mathcal{P}$ is NP-complete.

Corollary Let $\mathcal{P} = (\Pi, \Delta)$ be a DeLP-program such that for all $r \in (\Pi \cup \Delta)$, $|body(r)| \leq 2$. Deciding whether there is an argument for $\alpha \in Lit_{\mathcal{P}}$ w.r.t. $\mathcal{P}$ is NP-complete.

Proposition Given $\mathcal{P} = (\Pi, \Delta)$ and a literal $\alpha \in Lit_{\mathcal{P}}$, deciding whether there is an argument for α w.r.t. $\mathcal{P}$ is in PTIME if either *(i)* α does not depend in $G(\mathcal{P})$ on literals β and γ such that $\{\beta, \gamma\} \cup \Pi$ is contradictory, or *(ii)* α is not in $G(\mathcal{P})$.

Corollary Let $\mathcal{P} = (\Pi, \Delta)$ be a DeLP-program such that for all $r \in (\Pi \cup \Delta)$, $|body(r)| \leq 2$. Deciding whether $S_{\mathcal{P}}(\alpha) = \text{IN}$, $S_{\mathcal{P}}(\alpha) = \text{OUT}$, or $S_{\mathcal{P}}(\alpha) = \text{UNDECIDED}$, for $\alpha \in Lit_{\mathcal{P}}$ is NP-hard.

Outline

Updating a DeLP program

- An update consists of modifying a DeLP-program $\mathcal{P}$ into a new DeLP-program $\mathcal{P}^{'}$ by adding or removing a strict or a defeasible rule.

Example (Perform $u = +(\sim i \prec h)$ on $\mathcal{P}_1$)

The updated DeLP-program $\mathcal{P}_1' = (\Pi_1', \Delta_1')$, is as follows:

$$\Pi_1' = \Pi_1 = \{\sim a,\ t,\ b,\ (d \leftarrow t)\}$$

$$\Delta_1' = \Delta_1 = \left\{\begin{array}{lll} (i \prec s), & (s \prec h), & (h \prec b), \\ (\sim h \prec d, t), & (\sim i \prec \sim a, s), & (a \prec t), \\ (s \prec d), & (h \prec d), & (\sim f \prec \sim e), \\ (\sim e \prec \sim h, \sim a) & & \end{array}\right\} \cup \{(\sim i \prec h)\}$$

- If r is a strict rule and $u = +r$, then $\mathcal{P}' = ((\Pi \cup \{r\}), \Delta)$ if $(\Pi \cup \{r\})$ is guaranteed to be not contradictory, otherwise $\mathcal{P}' = \mathcal{P}$.

Question

- After performing an update the conclusion that can be derived may change.

Should we recompute the status of literals from scratch?

- The fact that computing the status of arguments is hard motivated the investigation of incremental techniques.

Overview of our incremental approach

Two main steps:

1) First, we check if the update is *irrelevant* (the status of all literals are preserved). In such a case we simply return the initial status $S_{\mathcal{P}}$.

2) To efficiently deal with *relevant* updates, we identify the subset of literals whose status needs to be recomputed after performing an update, and only recompute their status.

Hyper-graph for a DeLP-Program

Given a program $\mathcal{P}$, $G(\mathcal{P}) = \langle N, H \rangle$ is defined as follows:

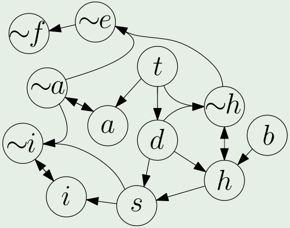
- If there is a strict derivation in Π for literal α, then $\alpha \in N$;
- For each strict rule $\alpha_0 \leftarrow \alpha_1, \ldots, \alpha_n$ (resp., defeasible rule $\alpha_0 \prec \alpha_1, \ldots, \alpha_n$) such that $\alpha_1, \ldots, \alpha_n \in N$, then $\alpha_0 \in N$ and $(\{\alpha_1, \ldots, \alpha_n\}, \alpha_0) \in H$;
- For each pair of nodes in N representing complementary literals α and $\sim\alpha$, both $(\{\alpha\}, \sim\alpha) \in H$ and $(\{\sim\alpha\}, \alpha) \in H$.

Example (Hyper-graph $G(\mathcal{P}_1)$ for $\mathcal{P}_1$)

Consider the DeLP-program $\mathcal{P}_1 = (\Pi_1, \Delta_1)$, where:

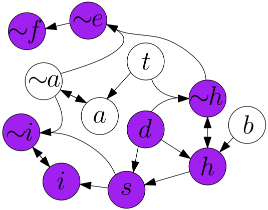
$\Pi_1 = \{\sim a, t, b, (d \leftarrow t)\}$

$$\Delta_1 = \left\{ \begin{array}{lll} (i \prec s), & (s \prec h), & (h \prec b), \\ (\sim h \prec d, t), & (\sim i \prec \sim a, s), & (a \prec t), \\ (s \prec d), & (h \prec d), & (\sim f \prec \sim e), \\ (\sim e \prec \sim h, \sim a) & & \end{array} \right\}$$



Reachable and Preserved Literals

- We say that a node y is *reachable* from a set X of nodes if there exists a hyper-path from X to y.
- We use $Reach_{G(\mathcal{P})}(X)$ to denote the set of all nodes that are reachable from X in $G(\mathcal{P})$.
- $Reach_{G(\mathcal{P}_1)}(\{d\}) = \{d, h, {\sim}h, s, {\sim}i, i, {\sim}e, {\sim}f\}$



Lemma (1) (Preserved literals)

Let $\mathcal{P}$ be a DeLP-program, $u = \pm r$ an update for $\mathcal{P}$, and $\mathcal{R}(u,\mathcal{P}) = Reach_{G(u,\mathcal{P})}(\{head(r)\})$. Let $\mathcal{P}' = u(\mathcal{P})$ be the updated program, and $G(\mathcal{P}') = \langle N', H' \rangle$ be the updated hyper-graph. Then, a literal $\alpha \in N'$ is preserved (i.e., $S_{\mathcal{P}}(\alpha) = S_{\mathcal{P}'}(\alpha)$) if $\alpha \notin \mathcal{R}(u,\mathcal{P})$.

- If a literal is not reachable in the hyper-graph, then its status does not change after the update.

Irrelevant updates

Proposition (*(2) Status of the head of the rule*)

Let $\mathcal{P}$ be a DeLP-program and $r = \alpha_0 \prec \alpha_1, \ldots, \alpha_n$ a ***defeasible*** *rule such that $\{\alpha_0, \ldots, \alpha_n\} \subseteq (Lit_{\mathcal{P}} \cap Lit_{\mathcal{P}'})$.*

(1) If $S_{\mathcal{P}}(\alpha_0) = \text{IN}$ then $+r$ is irrelevant for $\mathcal{P}$.

(2) If $S_{\mathcal{P}}(\alpha_0) = \text{OUT}$ then $-r$ is irrelevant for $\mathcal{P}$.

Proposition (*(3) Belonging to the Hyper-Graph*)

Let $\mathcal{P}$ be a DeLP-program and r a ***strict*** *rule $\alpha_0 \leftarrow \alpha_1, \ldots, \alpha_n$ or* ***defeasible*** *rule $\alpha_0 \prec \alpha_1, \ldots, \alpha_n$ such that $\{\alpha_0, \ldots \alpha_n\} \subseteq (Lit_{\mathcal{P}} \cap Lit_{\mathcal{P}'})$. Update $u = \pm r$ is irrelevant for $\mathcal{P}$ if α_0 does not belong to $G(u, \mathcal{P})$.*

Proposition (*(4) Reachable in the Hyper-Graph*)

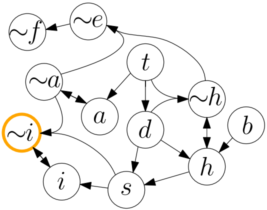
Let $\mathcal{P}$ be a DeLP-program and r a ***strict*** *rule $\alpha_0 \leftarrow \alpha_1, \ldots, \alpha_n$ or* ***defeasible*** *rule $\alpha_0 \prec \alpha_1, \ldots, \alpha_n$ such that $\{\alpha_0, \ldots \alpha_n\} \subseteq (Lit_{\mathcal{P}} \cap Lit_{\mathcal{P}'})$. Update $u = \pm r$ is irrelevant for $\mathcal{P}$ if there is α_i (with $i \in [1..n]$) such that $S_{\mathcal{P}}(\alpha_i) = \text{OUT}$ and $\alpha_i \notin Reach_{G(u,\mathcal{P})}(\{\alpha_0\})$.*

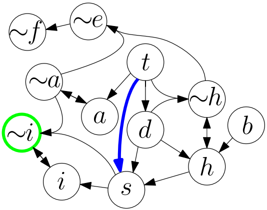
Relevant updates

- **However, in many cases updates are not irrelevant.**
- An update is relevant whenever it causes the status of at least one literal to change.

Example (A relevant update)

Consider again $\mathcal{P}_1$, where we have that $S_{\mathcal{P}_1}(s) = S_{\mathcal{P}_1}(t) = \text{IN}$. For update $u = +(s \leftarrow t)$, we have that $S_{u(\mathcal{P}_1)}(\sim i) = \text{IN}$, though it was UNDECIDED before performing the update. The change in the status of s is caused by the new argument $\langle \mathcal{A}_{10}, \sim i \rangle = \langle \{(\sim i \prec \sim a, s)\}, \sim i \rangle$ for $u(\mathcal{P}_1)$ and $\mathcal{A}_{10}$ is preferred to all the other arguments of the form $\langle \mathcal{A}, i \rangle$.





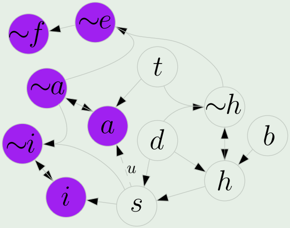
Influenced Set

We propose the concept of *influenced set*, which consists of the literals that are reachable in $G(u, \mathcal{P})$ from the head of the rule r in the update u by using only the hyper-edges whose body does not contain an unreachable literal whose status is OUT.

Example (Influenced literals)

Consider the update $u = +(a \prec s)$ over $\mathcal{P}_1$, which yields the DeLP-program $u(\mathcal{P}_1)$. Thus, we have:
$\mathcal{R}(u, \mathcal{P}_1) = \{a, \sim a, i, \sim i, \sim e, \sim f\}$, and

$\mathcal{R}(u, \mathcal{P}_1) \supseteq \mathcal{I}(u, \mathcal{P}_1, S_{\mathcal{P}_1}) = \{a, \sim a, i, \sim i\}$.



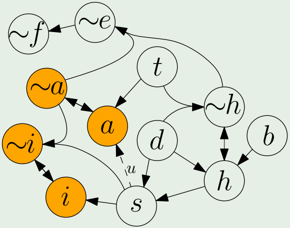
Influenced Set

We propose the concept of *influenced set*, which consists of the literals that are reachable in $G(u, \mathcal{P})$ from the head of the rule r in the update u by using only the hyper-edges whose body does not contain an unreachable literal whose status is OUT.

Example (Influenced literals)

Consider the update $u = +(a \prec s)$ over $\mathcal{P}_1$, which yields the DeLP-program $u(\mathcal{P}_1)$. Thus, we have:
$\mathcal{R}(u, \mathcal{P}_1) = \{a, \sim a, i, \sim i, \sim e, \sim f\}$, and

$\mathcal{R}(u, \mathcal{P}_1) \supseteq \mathcal{I}(u, \mathcal{P}_1, S_{\mathcal{P}_1}) = \{a, \sim a, i, \sim i\}$.



Influenced Set: Definition

We propose the concept of *influenced set*, which consists of the literals that are reachable in $G(u, \mathcal{P})$ from the head of the rule r in the update u by using only the hyper-edges whose body does not contain an unreachable literal whose status is OUT.

Definition (Influenced Set)

Let $\mathcal{P}$ be a DeLP-program, $u = \pm r$, and $S_{\mathcal{P}}$ the status of literals w.r.t. $\mathcal{P}$, and $G(u, \mathcal{P}) = \langle N^u, H^u \rangle$.

– $\mathcal{I}_0(u, \mathcal{P}, S_{\mathcal{P}}) = \begin{cases} \emptyset & \textit{if } u \textit{ is irrelevant for } \mathcal{P} \\ \{head(r)\} & \textit{otherwise}; \end{cases}$

– $\mathcal{I}_{i+1}(u, \mathcal{P}, S_{\mathcal{P}}) = \mathcal{I}_i(u, \mathcal{P}, S_{\mathcal{P}}) \cup \{\sim\alpha \mid \exists(\{\alpha\}, \sim\alpha) \in H^u \textit{ s.t. } \alpha \in \mathcal{I}_i(u, \mathcal{P}, S_{\mathcal{P}})\} \cup \{y \mid \exists(X, \alpha) \in H^u \textit{ s.t. } X \cap \mathcal{I}_i(u, \mathcal{P}, S_{\mathcal{P}}) \neq \emptyset \wedge X \cap OUT(u, \mathcal{P}, S_{\mathcal{P}}) = \emptyset\}$.

The *influenced set* for u w.r.t. $\mathcal{P}$ and $S_{\mathcal{P}}$ is then defined as $\mathcal{I}(u, \mathcal{P}, S_{\mathcal{P}}) = \mathcal{I}_n(u, \mathcal{P}, S_{\mathcal{P}})$ such that $\mathcal{I}_n(u, \mathcal{P}, S_{\mathcal{P}}) = \mathcal{I}_{n+1}(u, \mathcal{P}, S_{\mathcal{P}})$.

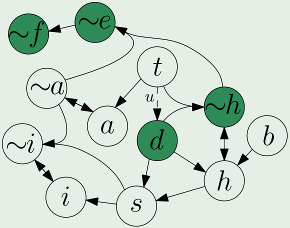
Inferable and Core Literals

[Inferable] The status of a literal for which there is no argument in the (updated) program may depend only on the status of its complementary literal—we call such literals *inferable*.

[Core] The core literals for a relevant update $u = \pm r$ w.r.t. $\mathcal{P}$ are those in $Lit_{\mathcal{P}'}$ that are influenced but are not inferable.

Example (Inferbale and Core literals)

Consider the update $u = -(d \leftarrow t)$ over $\mathcal{P}_1$, which yields the DeLP-program $u(\mathcal{P}_1)$. Thus, we have:



$Infer(u, \mathcal{P}_1) = \{d, \sim h, \sim e, \sim f\}$

$Core(u, \mathcal{P}_1) = \{h, s, \sim i, i\}$.

Inferable and Core Literals

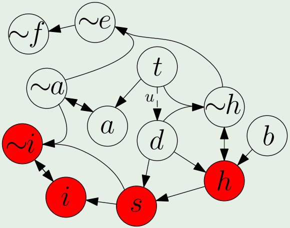
[Inferable] The status of a literal for which there is no argument in the (updated) program may depend only on the status of its complementary literal—we call such literals *inferable*.

[Core] The core literals for a relevant update $u = \pm r$ w.r.t. $\mathcal{P}$ are those in $Lit_{\mathcal{P}'}$ that are influenced but are not inferable.

Example (Inferbale and Core literals)

Consider the update $u = -(d \leftarrow t)$ over $\mathcal{P}_1$, which yields the DeLP-program $u(\mathcal{P}_1)$. Thus, we have:

$Infer(u, \mathcal{P}_1) = \{d, \sim h, \sim e, \sim f\}$
$Core(u, \mathcal{P}_1) = \{h, s, \sim i, i\}$.



Inferable and Core Literals: Definitions

The status of a literal for which there is no argument in the (updated) program may depend only on the status of its complementary literal—we call such literals *inferable*. Using the hyper-graph of updated programs, we can define inferable literals as follows.

Definition (Set of Inferable Literals)

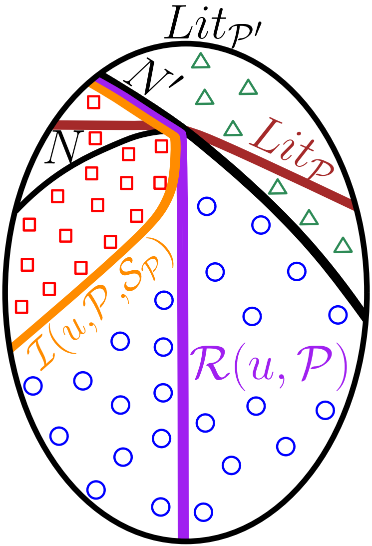
Let $\mathcal{P}$ be a DeLP-program, $u = \pm r$, $\mathcal{P}' = u(\mathcal{P})$, and $G(\mathcal{P}') = \langle N', H' \rangle$. The set of inferable literals for u w.r.t. $\mathcal{P}$ is $\mathit{Infer}(u, \mathcal{P}) = \mathit{Lit}_{\mathcal{P}'} \setminus N'$.

The core literals for a relevant update $u = \pm r$ w.r.t. $\mathcal{P}$ are those in $\mathit{Lit}_{\mathcal{P}'}$ that are influenced but are not inferable.

Definition (Set of Core Literals)

Let $\mathcal{P}$ be a DeLP-program, $u = \pm r$, and $S_{\mathcal{P}}$ the status of the literals of $\mathcal{P}$. The set $\mathit{Core}(u, \mathcal{P})$ of core literals for u w.r.t. $\mathcal{P}$ is
$\mathit{Core}(u, \mathcal{P})) = (\mathcal{I}(u, \mathcal{P}, S_{\mathcal{P}}) \setminus \mathit{Infer}(u, \mathcal{P})) \cap \mathit{Lit}_{\mathcal{P}'}$.

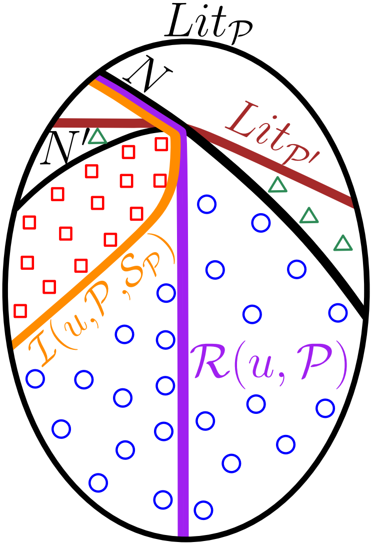
Relationships for addition



$Legend:$

○	PR
□	$Core(u,\mathcal{P})$
△	$Infer(u,\mathcal{P})$

Relationships for deletion



$Legend:$

○	PR
□	$Core(u,\mathcal{P})$
△	$Infer(u,\mathcal{P})$

Incremental Algorithm

Algorithm Dynamic DeLP-Solver

Input: DeLP-program $\mathcal{P}$, Initial status $S_{\mathcal{P}}$, Update $u = \pm r$.
Output: Status $S_{\mathcal{P}'}$ w.r.t. the updated program $\mathcal{P}' = u(\mathcal{P})$.

if one of Propositions 2–4 holds (the update is irrelevant) **then**
  return $S_{\mathcal{P}}$;// Nothing changes
if $\{head(r), \sim head(r)\} \cap Lit_{\mathcal{P}} = \emptyset$ **then**
  return $S_{\mathcal{P}} \cup \{(head(r), \text{DELP-SOLVER}(\mathcal{P}', head(r)))\}$;// New fresh literal
Let $G(\mathcal{P}') = \langle N', H' \rangle$;// Build the Hyper-Graph
Let $PR = \{\alpha \in N' \setminus \mathcal{I}(u, \mathcal{P}, S_{\mathcal{P}})\}$;// Preserved Literals
for $\alpha \in PR$ **do**
  $S_{\mathcal{P}'}(\alpha) \leftarrow S_{\mathcal{P}}(\alpha)$;// Status Preserved
for $\alpha \in Core(u, \mathcal{P})$ **do**
  $S_{\mathcal{P}'}(\alpha) \leftarrow \text{DELP-SOLVER}(\mathcal{P}', \alpha)$;// Status must be computed
for $\alpha \in Infer(u, \mathcal{P})$ **do**
  if $S_{\mathcal{P}'}(\sim\alpha) = \text{IN}$
    then $S_{\mathcal{P}'}(\alpha) \leftarrow \text{OUT}$// Status Inferred
  else $S_{\mathcal{P}'}(\alpha) \leftarrow \text{UNDECIDED}$;// Status Inferred
for $\alpha \in Lit \setminus Lit_{\mathcal{P}'}$ **do**
  $S_{\mathcal{P}'}(\alpha) = \text{UNKNOWN}$// Literal is not in the language of the updated program
return $S_{\mathcal{P}'}$.

Outline

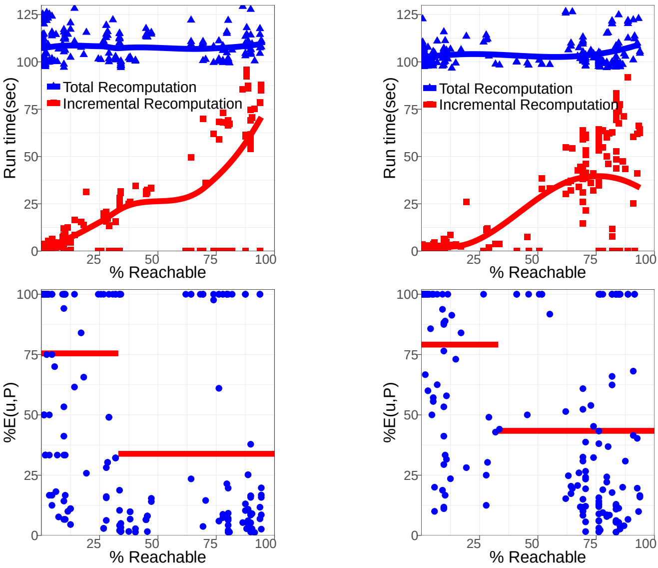
Dataset & Metodology

Datasets:
Inspired by the structure of the DeLP-program in our running example, we generated a set of 40 DeLP programs, each consisting of a number of literals in $\{180, 220\}$, of facts in $\{10, 20\}$, of strict rules in $\{20, 30\}$, and a number of defeasible rules in $\{100, 150\}$. For each program, we generated 5 different rule addition/deletion updates.

Methodology
[**Efficiency**] For each DeLP-program $\mathcal{P}$ in the dataset, we compared the average running time of Algorithm 1 with that of the approach from scratch, which computes the status in the updated program by directly calling the DeLP-Solver for each literal of $\mathcal{P}$.
[**Effectiveness**] We also measured the percentage of literals whose status needs to be recomputed over the set of literals whose status is recomputed by Algorithm 1.

$$E(u, \mathcal{P}) = \frac{|Rec(u, \mathcal{P})|}{|Core(u, \mathcal{P}) \cup Infer(u, \mathcal{P})|}$$

Experimental Results for addition/deletion (left/right)



Results

1) We compared our technique with the computation from scratch.
2) We performed experiments that aimed at evaluating both the efficiency and effectiveness of our approach.
3) Our incremental algorithm outperforms the computation from scratch.
4) For almost half of the updates performed, the proposed technique computes only the status of literals whose status actually needs to be recomputed.

Outline

* We have taken the first steps in tackling the problem of avoiding wasted effort when determining the warrant status of literals in a DeLP program after that a (defeasible or strict) rule is added/removed.

* Our incremental approach outperforms the computation from scratch (especially if the average number of literals reachable from an update is less than 33%).

FW1) Further developing these techniques, as well as developing similar ones for fact addition and deletion, and the more general case of simultaneously adding or deleting a *set* of rules and facts.

FW2) We believe the basic ideas in the framework could carry over to other frameworks, v.g. ASPIC+, ABA.

Thank you!

... any ~~question~~ argument?