Operator-based semantics for choice programs: is choosing losing?

Jesse Heyninck

November 5, 2024

Open Universiteit, the Netherlands
University of Cape Town, South-Africa

Motivation and Contributions

```
#count{attend(asp),attend(krl),attend(cex)}=1
:- #sum{-1:freshOutside, 1:not tired}=>0.
```

- Provide an operator-based view [**H**. et al., 2024] on semantics for choice programs,
- allowing to derive three-valued supported, stable and well-founded semantics in a principled way.
- Define a new notion of stable semantics for non-deterministic approximation fixpoint theory.
- Compare resulting semantics using various notions of groundedness.
- Give a clear view on the difference with DLPs.

Plan

Preliminaries on Choice Rules

Immediate Consequence Operators

Three-Valued Operators and Fixpoints

Stable Semantics

Groundedness

Disjunctions are Choice Constructs

Preliminaries on Choice Rules

Choice Atoms

A *choice atom* is an expression $C = (\mathsf{dom}, \mathsf{sat})$ where $\mathsf{dom} \subseteq \mathcal{A}$ and $\mathsf{sat} \subseteq \wp(\mathsf{dom})$.

Example

$1\{p, q, r\}2$ *corresponds to the choice atom*
$C_1 = (\{p, q, r\}, \{\{p\}, \{q\}, \{r\}, \{p, q\}, \{p, r\}, \{q, r\}\})$.

Choice Atoms

A *choice atom* is an expression $C = (\mathsf{dom}, \mathsf{sat})$ where $\mathsf{dom} \subseteq \mathcal{A}$ and $\mathsf{sat} \subseteq \wp(\mathsf{dom})$.

A set of atoms x satisfies $(\mathsf{dom}, \mathsf{sat})$ if $x \cap \mathsf{dom} \in \mathsf{sat}$.

Example

$1\{p, q, r\}2$ *corresponds to the choice atom*
$C_1 = (\{p, q, r\}, \{\{p\}, \{q\}, \{r\}, \{p, q\}, \{p, r\}, \{q, r\}\})$.

Choice Atoms

A *choice atom* is an expression $C = (\mathsf{dom}, \mathsf{sat})$ where $\mathsf{dom} \subseteq \mathcal{A}$ and $\mathsf{sat} \subseteq \wp(\mathsf{dom})$.

A set of atoms x satisfies $(\mathsf{dom}, \mathsf{sat})$ if $x \cap \mathsf{dom} \in \mathsf{sat}$.

Example

$1\{p, q, r\}2$ *corresponds to the choice atom* $C_1 = (\{p, q, r\}, \{\{p\}, \{q\}, \{r\}, \{p, q\}, \{p, r\}, \{q, r\}\})$.

- $\{p, q, s\}$ *satisfies* C_1 *as* $\{p, q, s\} \cap \{p, q, r\} = \{p, q\} \in \{\{p\}, \{q\}, \{r\}, \{p, q\}, \{p, r\}, \{q, r\}\}$.
- $\{p, q, r\}$ *does not satisfy* C_1 *as* $\{p, q, r\} \cap \{p, q, r\} = \{p, q, r\} \notin \{\{p\}, \{q\}, \{r\}, \{p, q\}, \{p, r\}, \{q, r\}\}$.

Choice Rules

Where C, $C_1, \ldots, C_n$ are choice atoms, a choice rule is of the form:

$$C \leftarrow C_1, \ldots, C_n.$$

Choice Rules

Where C, $C_1, \ldots, C_n$ are choice atoms, a choice rule is of the form:

$$C \leftarrow C_1, \ldots, C_n.$$

A program with choice rules is a set of choice rules.

Example
$\{1\{p, q\}2 \leftarrow \{p, q\} \neq 1\}$

Choice Rules

Where C, $C_1, \ldots, C_n$ are choice atoms, a choice rule is of the form:

$$C \leftarrow C_1, \ldots, C_n.$$

A program with choice rules is a set of choice rules.

Example
$\{1\{p, q\}2 \leftarrow \{p, q\} \neq 1\}$

- A choice rule is normal if $\text{sat}(C_i) = \{\{a\}\}$ (for some $a \in \mathcal{A}$) or $\text{sat}(C_i) = \{\emptyset\}$ for every $i = 1 \ldots n$,
  - $(\{a\}, \{\{a\}\})$ is denoted by a, and
  - $(\{a\}, \{\emptyset\})$ is denoted by `not` a.

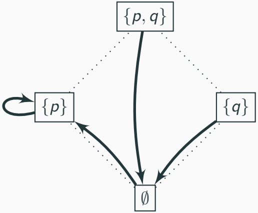
Immediate Consequence Operators

Immediate Consequence Operator: normal logic programs

$$IC_{\mathcal{P}}(x) = \{a \mid \quad a \leftarrow b_1, \ldots, b_n, \mathtt{not}\ c_1, \ldots, \mathtt{not}\ c_m \in \mathcal{P}, \text{ and } b_1, \ldots, b_n \in x, \text{ and } c_1, \ldots, c_m \notin x\}$$

Example

$\mathcal{P} = \{p \leftarrow \mathtt{not}\ q\}$

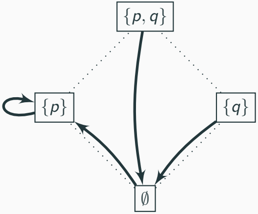


Immediate Consequence Operator: normal logic programs

$$IC_{\mathcal{P}}(x) = \{a \mid \quad a \leftarrow b_1, \ldots, b_n, \texttt{not}\ c_1, \ldots, \texttt{not}\ c_m \in \mathcal{P}, \text{ and } b_1, \ldots, b_n \in x, \text{ and } c_1, \ldots, c_m \notin x\}$$

Example

$\mathcal{P} = \{p \leftarrow \texttt{not}\ q\}$



Interested in (selected) fixpoints as semantics of logic programs.

Constructive definition of semantics.

Only job: specify an operator.

Non-Deterministic Immediate Consequence Operator

Given a choice program $\mathcal{P}$ and a set of atoms x:

- A rule $r \in \mathcal{P}$ is x-applicable (in symbols, $r \in \mathcal{P}(x)$) if x satisfies the body of r.
- $IC_{\mathcal{P}}(x) = \{z \subseteq \bigcup_{r \in \mathcal{P}(x)} \mathrm{dom}(\mathrm{hd}(r)) \mid \forall r \in \mathcal{P}(x) : z \models \mathrm{hd}(r)\}$

Non-Deterministic Immediate Consequence Operator

Given a choice program $\mathcal{P}$ and a set of atoms x:

- A rule $r \in \mathcal{P}$ is x-applicable (in symbols, $r \in \mathcal{P}(x)$) if x satisfies the body of r.
- $IC_{\mathcal{P}}(x) = \{z \subseteq \bigcup_{r \in \mathcal{P}(x)} \mathrm{dom}(\mathrm{hd}(r)) \mid \forall r \in \mathcal{P}(x) : z \models \mathrm{hd}(r)\}$
  All ways to make the heads of x-applicable rules true.

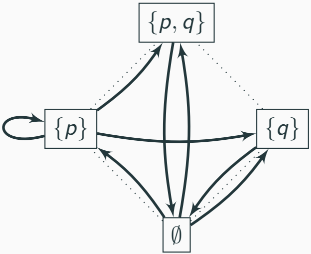
Non-Deterministic Immediate Consequence Operator

Given a choice program $\mathcal{P}$ and a set of atoms x:

- A rule $r \in \mathcal{P}$ is x-applicable (in symbols, $r \in \mathcal{P}(x)$) if x satisfies the body of r.
- $IC_{\mathcal{P}}(x) = \{z \subseteq \bigcup_{r \in \mathcal{P}(x)} \mathrm{dom}(\mathrm{hd}(r)) \mid \forall r \in \mathcal{P}(x) : z \models \mathrm{hd}(r)\}$
  All ways to make the heads of x-applicable rules true.

Example

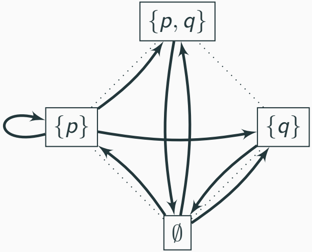
$\mathcal{P} = \{1\{p, q\}2 \leftarrow \texttt{not } q\}$



Proposition
Let a choice program $\mathcal{P}$ be given. Then $x \in IC_{\mathcal{P}}(x)$ iff x is a supported model of $\mathcal{P}$ according to [Liu et al., 2010].

Example $\mathcal{P} = \{1\{p, q\}2 \leftarrow \mathtt{not}\ q\}$

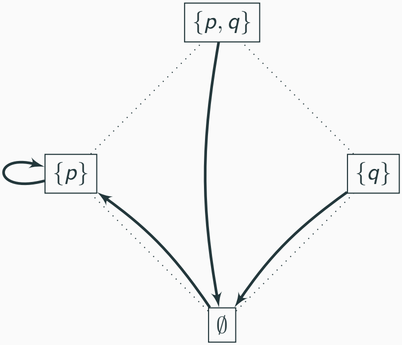
$\{p\}$ is a fixpoint
as $\{\{p\}, \{q\}, \{p, q\}\} = IC_{\mathcal{P}}(\{p\})$.



Three-Valued Operators and Fixpoints

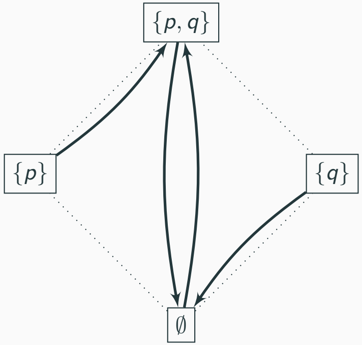
Non-Monotonic Operators

$\mathcal{P} = \{p \leftarrow \texttt{not}\ q\}$

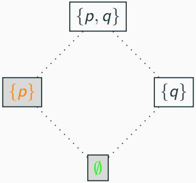


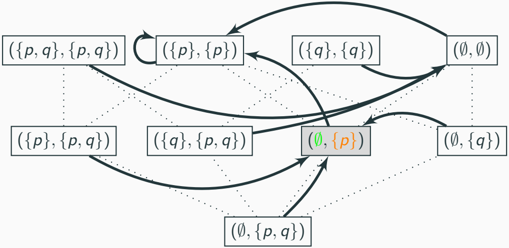
Non-Monotonic Operators

$$\mathcal{P} = \{p \leftarrow \texttt{not}\ q., \quad q \leftarrow \texttt{not}\ p.\}$$



Non-Monotonic Operators





- Intervals (pairs of sets) approximate single sets.
- Information ordering $\leq_i$ between intervals.
- $\leq_i$-monotonic operator $\mathcal{IC}_{\mathcal{P}}$ approximates original operator $IC_{\mathcal{P}}$.

Non-Deterministic Approximators for Normal Choice Programs

$$\mathcal{HD}^l_{\mathcal{P}}(x, y) = \{C \mid C \leftarrow a_1, \ldots, a_n, \texttt{not}\ b_1, \ldots, \texttt{not}\ b_m \in \mathcal{P},$$
$$\{a_1, \ldots, a_n\} \subseteq x, \{b_1, \ldots, b_m\} \cap y = \emptyset\}\}$$

$$\mathcal{IC}^l_{\mathcal{P}}(x, y) = \{z \subseteq \bigcup_{C \in \mathcal{HD}^l_{\mathcal{P}}(x,y)} \mathsf{dom}(C) \mid \forall C \in \mathcal{HD}^l_{\mathcal{P}}(x, y), z \cap \mathsf{dom}(C) \in \mathsf{sat}(C)\}$$

$$\mathcal{IC}^u_{\mathcal{P}}(x, y) = \mathcal{IC}^l_{\mathcal{P}}(y, x)$$

$$\mathcal{IC}_{\mathcal{P}}(x, y) = (\mathcal{IC}^l_{\mathcal{P}}(x, y), \mathcal{IC}^u_{\mathcal{P}}(x, y))$$

Non-Deterministic Approximators for Normal Choice Programs

$$\mathcal{HD}^{l}_{\mathcal{P}}(x,y) = \{C \mid C \leftarrow a_1,\ldots,a_n, \texttt{not}\ b_1,\ldots,\texttt{not}\ b_m \in \mathcal{P},$$
$$\{a_1,\ldots,a_n\} \subseteq x, \{b_1,\ldots,b_m\} \cap y = \emptyset\}\}$$

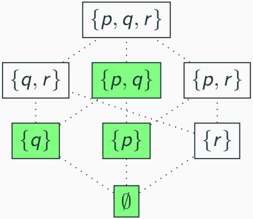
$$\mathcal{IC}^{l}_{\mathcal{P}}(x,y) = \{z \subseteq \bigcup_{C \in \mathcal{HD}^{l}_{\mathcal{P}}(x,y)} \mathrm{dom}(C) \mid \forall C \in \mathcal{HD}^{l}_{\mathcal{P}}(x,y), z \cap \mathrm{dom}(C) \in \mathrm{sat}(C)\}$$

$$\mathcal{IC}^{u}_{\mathcal{P}}(x,y) = \mathcal{IC}^{l}_{\mathcal{P}}(y,x)$$

$$\mathcal{IC}_{\mathcal{P}}(x,y) = (\mathcal{IC}^{l}_{\mathcal{P}}(x,y), \mathcal{IC}^{u}_{\mathcal{P}}(x,y))$$

Example: $\mathcal{P} = \{1\{p,r\}2 \leftarrow \texttt{not}\ r; p \leftarrow q; q \leftarrow p\}$

- $\mathcal{HD}^{c,l}_{\mathcal{P}}(\emptyset, \{r\}) = \emptyset$,
- $HDc^{c,u}_{\mathcal{P}}(\emptyset, \{r\}) = \{1\{p,q\}2\}$,
- $\mathcal{IC}^{c}_{\mathcal{P}}(\emptyset, \{r\}) = (\{\emptyset\}, \{\{p\},\{q\},\{p,q\}\})$.



Approximators for Choice Programs (aka the real fun)

Given a choice program $\mathcal{P}$ and pair of sets of atoms (x, y) let:

$$\mathcal{HD}^{\mathsf{GZ},l}_{\mathcal{P}}(x,y) = \{C \mid \exists C \leftarrow C_1, \ldots, C_i \in \mathcal{P}, \forall i = 1 \ldots n :$$
$$x \cap \mathrm{dom}(C_i) = y \cap \mathrm{dom}(C_i) \in \mathrm{sat}(C_i)\},$$
$$\mathcal{HD}^{\mathsf{LPST},l}_{\mathcal{P}}(x,y) = \{C \mid \exists C \leftarrow C_1, \ldots, C_n \in \mathcal{P}, \forall i = 1 \ldots n :$$
$$\forall z \in [x,y] : z(C_i) = \mathsf{T}\},$$
$$\mathcal{HD}^{\mathsf{MR},l}_{\mathcal{P}}(x,y) = \{C \mid \exists C \leftarrow C_1, \ldots, C_n \in \mathcal{P}, \exists z \subseteq x :$$
$$\forall i = 1 \ldots n : y(C_i) = \mathsf{T} \text{ and } z(C_i) = \mathsf{T}\},$$

For $\mathsf{x} \in \{\mathsf{LPST}, \mathsf{MR}, \mathsf{GZ}\})$ let:

$$\mathcal{IC}^{\mathsf{x},l}(x,y) = \{z \subseteq \bigcup_{C \in \mathcal{HD}^{\mathsf{x},l}_{\mathcal{P}}(x,y)} \mathrm{dom}(C) \mid \forall C \in \mathcal{HD}^{\mathsf{x},l}_{\mathcal{P}}(x,y) : z \cap \mathrm{dom}(C) \in \mathrm{sat}(C)\}$$

For $\dagger \in \{\mathsf{MR}, \mathsf{LPST}, \mathcal{U}\}$ let:

$$\mathcal{IC}^{\mathcal{U},l}_{\mathcal{P}}(x,y) = \mathcal{IC}^{\dagger,u}_{\mathcal{P}}(x,y) \qquad = \bigcup_{x \subseteq z \subseteq y} IC_{\mathcal{P}}(z)$$

whereas $\mathcal{IC}^{\mathsf{GZ},u}_{\mathcal{P}}(x,y) = \mathcal{IC}^{\mathsf{GZ},l}_{\mathcal{P}}(x,y)$. $\mathcal{IC}^{\mathsf{x}}(x,y) = (\mathcal{IC}^{\mathsf{x},l}(x,y), \mathcal{IC}^{\mathsf{x},u}(x,y))$ (for $\mathsf{x} \in \{\mathsf{LPST}, \mathsf{MR}, \mathsf{GZ}, \mathcal{U}\}$).

Non-Deteterministic AFT: fixpoints

Our notion generalizes supported models to the three-valued case:

Example $\mathcal{P} = \{\{p, q\} = 1 \leftarrow \texttt{not}\, p. \quad p \leftarrow q.\}$

- No (two-valued) supported models

Non-Deteterministic AFT: fixpoints

Our notion generalizes supported models to the three-valued case:

Example $\mathcal{P} = \{\{p, q\} = 1 \leftarrow \texttt{not}\ p. \quad p \leftarrow q.\}$

- No (two-valued) supported models
- $\mathcal{IC}^c_{\mathcal{P}}(\emptyset, \{p\}) = (\{\emptyset\}, \{\{p\}, \{q\}, \{p, q\}\})$.
- (among others)

Proposition
Let a normal logic program $\mathcal{P}$ be given. Then $(x, y) \in \mathcal{IC}_{\mathcal{P}}(x, x)$ iff (x, y) is a partial supported model of $\mathcal{P}$.

Proposition
Let a choice program $\mathcal{P}$ and $\mathsf{x} \in \{\mathrm{LPST}, \mathrm{MR}, \mathrm{GZ}, \mathcal{U}\}$ be given. If $(x, y) \in \mathcal{IC}^{\mathsf{x}}_{\mathcal{P}}(x, y)$ then for every $a \in y$, there is a $C \leftarrow C_1, \ldots, C_n$ with $a \in \mathrm{dom}(C)$ s.t. $\mathcal{IC}^{\mathsf{x}}_{\{p \leftarrow C_1, \ldots, C_n\}}(x, y) = (\{\{p\} \cap x\}, \{\{p\} \cap y\})$.

Stable Semantics

Stable Operator for Deterministic Approximation Operators

Avoid self-supporting conclusions by fixing what is false and accepting only what is necessary in view thereof:

$$S(\mathcal{IC}^l_{\mathcal{P}})(y) = glb\{x \subseteq \mathcal{A} \mid x = \mathcal{IC}^l_{\mathcal{P}}(x, y)\} = \bigcup_{i=1}^{\infty} (\mathcal{IC}^l_{\mathcal{P}})^i(\emptyset, y).$$

Stable Operator for Deterministic Approximation Operators

Avoid self-supporting conclusions by fixing what is false and accepting only what is necessary in view thereof:

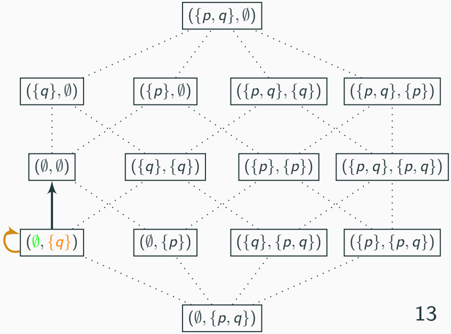
$$S(\mathcal{IC}^l_{\mathcal{P}})(y) = glb\{x \subseteq \mathcal{A} \mid x = \mathcal{IC}^l_{\mathcal{P}}(x, y)\} = \bigcup_{i=1}^{\infty} (\mathcal{IC}^l_{\mathcal{P}})^i(\emptyset, y).$$

Example

$\{p \leftarrow \texttt{not}\ q.; \quad q \leftarrow q.\}$

$S(\mathcal{IC}_{\mathcal{P}})(\{q\}, \{q\}) = (\emptyset, \emptyset)$.

- $\mathcal{IC}^l_{\mathcal{P}}(\emptyset, \{q\}) = \emptyset$.



Avoid self-supporting conclusions by fixing what is false and accepting only what is necessary in view thereof:

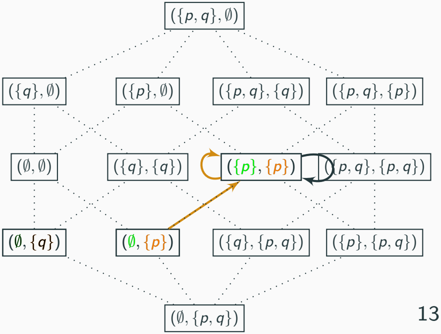
$$S(\mathcal{IC}^l_{\mathcal{P}})(y) = glb\{x \subseteq \mathcal{A} \mid x = \mathcal{IC}^l_{\mathcal{P}}(x, y)\} = \bigcup_{i=1}^{\infty} (\mathcal{IC}^l_{\mathcal{P}})^i(\emptyset, y).$$

Example

$\{p \leftarrow \texttt{not}\ q.; \quad q \leftarrow q.\}$

$S(\mathcal{IC}_{\mathcal{P}})(\{p\}, \{p\}) = (\{p\}, \{p\})$.

- $\mathcal{IC}_{\mathcal{P}}(\emptyset, \{p\}) = \{p\}$.
- $\mathcal{IC}_{\mathcal{P}}(\{p\}, \{p\}) = \{p\}$.



Constructive Stable Operators and Fixpoints

Given a non-deterministic operator $O : \mathcal{L} \to \wp(\mathcal{L})$, a sequence $x_0, \ldots, x_n \subseteq \mathcal{L}$ is well-founded relative to O if:

- $x_0 = \bot$,
- $x_i \leq x_{i+1}$ and $x_{i+1} \in O(x_i)$ for every successor ordinal $i \geq 0$.
- $x_\lambda = (lub\{x_i\}_{i<\lambda})$ for a limit ordinal λ.

Constructive Stable Operators and Fixpoints

Given a non-deterministic operator $O : \mathcal{L} \to \wp(\mathcal{L})$, a sequence $x_0, \ldots, x_n \subseteq \mathcal{L}$ is well-founded relative to O if:

- $x_0 = \bot$,
- $x_i \leq x_{i+1}$ and $x_{i+1} \in O(x_i)$ for every successor ordinal $i \geq 0$.
- $x_\lambda = (lub\{x_i\}_{i<\lambda})$ for a limit ordinal λ.

The complete constructive lower bound operator is defined as:

$$S^c(\mathcal{IC}^l_{\mathcal{P}})(y) = \{x \in \mathcal{O}_l(x,y) \mid \exists x_0, .., x \in \text{wfs}(\mathcal{IC}^l_{\mathcal{P}}(., y))\}$$

The complete constructive upper bound operator is defined analogously, and the constructive stable operator is defined as $S^c(\mathcal{IC}_{\mathcal{P}})(x,y) = (S^c(\mathcal{IC}^l_{\mathcal{P}})(y), S^c(\mathcal{IC}^u_{\mathcal{P}})(x))$.

Constructive Stable Operators and Fixpoints

Given a non-deterministic operator $O : \mathcal{L} \to \wp(\mathcal{L})$, a sequence $x_0, \ldots, x_n \subseteq \mathcal{L}$ is well-founded relative to O if:

- $x_0 = \bot$,
- $x_i \leq x_{i+1}$ and $x_{i+1} \in O(x_i)$ for every successor ordinal $i \geq 0$.
- $x_\lambda = (lub\{x_i\}_{i<\lambda})$ for a limit ordinal λ.

The complete constructive lower bound operator is defined as:

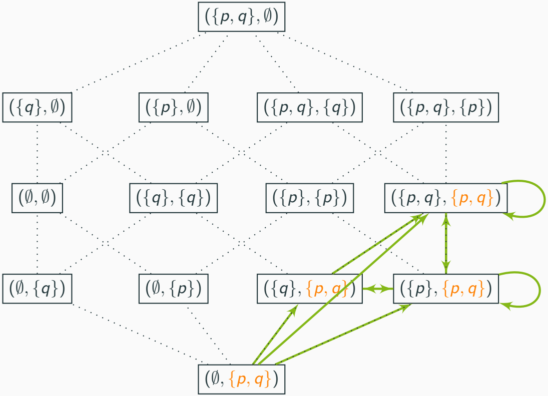
$$S^c(\mathcal{IC}^l_{\mathcal{P}})(y) = \{x \in \mathcal{O}_l(x, y) \mid \exists x_0, .., x \in \mathrm{wfs}(\mathcal{IC}^l_{\mathcal{P}}(., y))\}$$

The complete constructive upper bound operator is defined analogously, and the constructive stable operator is defined as $S^c(\mathcal{IC}_{\mathcal{P}})(x, y) = (S^c(\mathcal{IC}^l_{\mathcal{P}})(y), S^c(\mathcal{IC}^u_{\mathcal{P}})(x))$.
A pair (x, y) is a constructive stable fixpoint iff $(x, y) \in S^c(\mathcal{IC}_{\mathcal{P}})(x, y)$.

Stable Operator: Example 1

$$\mathcal{P} = \{1\{p,q\}2 \leftarrow .,\quad p \leftarrow q\}$$

y	$\emptyset$	$\{p\}$	$\{q\}$	$\{p,q\}$
$\mathcal{IC}^{c,l}_{\mathcal{P}}(.,y)$	$\{p\},\{q\},\{p,q\}$	$\{p\},\{q\},\{p,q\}$	$\{p\},\{p,q\}$	$\{p\},\{p,q\}$



$S(\mathcal{IC}^l_{\mathcal{P}})(\{p,q\}) =$
$\{\{p\},\{p,q\}\}$

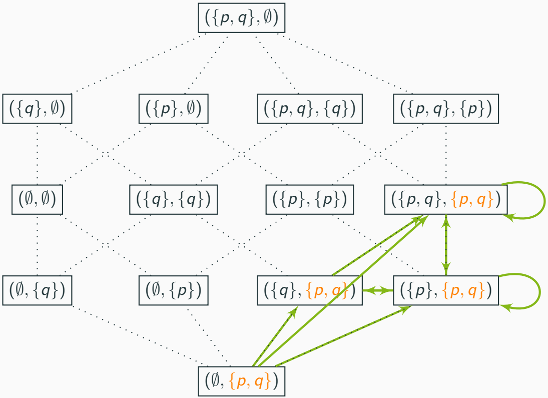
$(\{p\},\{p\})$ is a stable fp.

$(\{p,q\},\{p,q\})$ is a stable fp.

Stable Operator: Example 1

$$\mathcal{P} = \{1\{p,q\}2 \leftarrow .,\quad p \leftarrow q\}$$

y	$\emptyset$	$\{p\}$	$\{q\}$	$\{p,q\}$
$\mathcal{IC}^{c,l}_{\mathcal{P}}(.,y)$	$\{p\},\{q\},\{p,q\}$	$\{p\},\{q\},\{p,q\}$	$\{p\},\{p,q\}$	$\{p\},\{p,q\}$



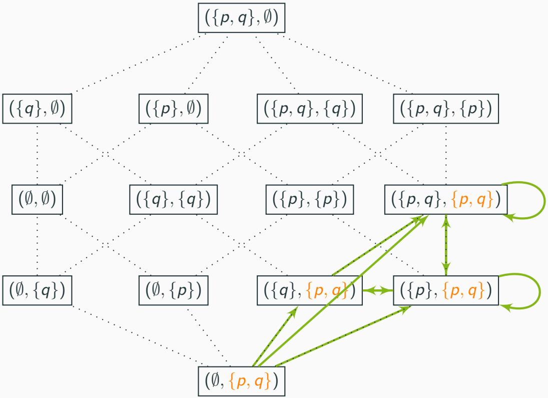
$S(\mathcal{IC}^{l}_{\mathcal{P}})(\{p,q\}) =$
$\{\{p\},\{p,q\}\}$

$(\{p\},\{p\})$ is a stable fp.

$(\{p,q\},\{p,q\})$ is a stable fp.

$$\mathcal{P} = \{1\{p, q\}2 \leftarrow .,\quad p \leftarrow q\}$$

y	$\emptyset$	$\{p\}$	$\{q\}$	$\{p, q\}$
$\mathcal{IC}^{c,l}_{\mathcal{P}}(., y)$	$\{p\}, \{q\}, \{p, q\}$	$\{p\}, \{q\}, \{p, q\}$	$\{p\}, \{p, q\}$	$\{p\}, \{p, q\}$



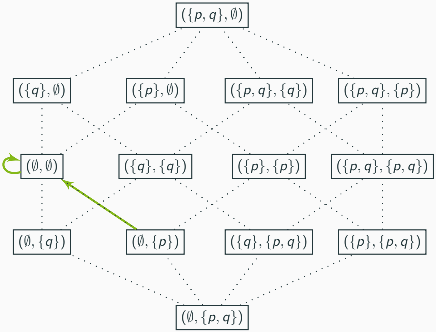
$S(\mathcal{IC}^{l}_{\mathcal{P}})(\{p, q\}) =$
$\{\{p\}, \{p, q\}\}$

$(\{p\}, \{p\})$ is a stable fp.

$(\{p, q\}, \{p, q\})$ is a stable fp.

Stable Operator: Example 2

$$\mathcal{P} = \{1\{p,q\} \leftarrow p.; \quad 1\{p,q\} \leftarrow q\}$$



Well-founded sequence of $IC^{l}_{\mathcal{P}}(.,y)$ for any $y \subseteq \{p,q\}$: $\emptyset$

$(\emptyset,\emptyset)$ is the unique fixpoint of $S(\mathcal{IC}_{\mathcal{P}})$.

General results and characterisation theorem

Theorem
Let a choice program $\mathcal{P}$ s.t. for every $C_1 \leftarrow C_2, \ldots, C_n \in \mathcal{P}$, $\mathrm{dom}(C_i)$ is finite for $i = 1 \ldots n$, and $\mathsf{x} \in \{\mathsf{MR}, \mathsf{LPST}, \mathcal{U}\}$ and $x \subseteq y \subseteq \mathcal{A}$ be given. Then $S^c(\mathcal{IC}^{\mathsf{x}}_{\mathcal{P}})(x, y) \neq \emptyset$. Furthermore, $C^c(\mathcal{IC}^{\mathsf{GZ},l}_{\mathcal{P}})(y) \neq \emptyset$.

Theorem
Let a choice program $\mathcal{P}$ be given.

1. *x is a stable model according to [Liu et al., 2010] iff (x, x) is a stable fixpoint of $\mathcal{IC}^{\mathsf{LPST}}_{\mathcal{P}}$.*
2. *x is a stable model according to [Marek and Remmel, 2004] iff (x, x) is a stable fixpoint of $\mathcal{IC}^{\mathsf{MR}}_{\mathcal{P}}$.*
3. *If $\mathcal{P}$ is a aggregate program then x is a stable model according to [Gelfond and Zhang, 2014] iff $x \in C^c(\mathcal{IC}^{\mathsf{GZ},l}_{\mathcal{P}})(x)$.*

Groundedness

Groundedness

A set x is grounded relative to $\mathcal{P}$ if there is a mapping $\kappa : x \to \mathbb{N}$ s.t. for every $a \in x$, there is an $a \leftarrow a_1, \ldots, a_n$, $\kappa(a) > \kappa(a_i)$ for every $i = 1 \ldots n$.

Groundedness

A set x is grounded relative to $\mathcal{P}$ if there is a mapping $\kappa : x \to \mathbb{N}$ s.t. for every $a \in x$, there is an $a \leftarrow a_1, \ldots, a_n$, $\kappa(a) > \kappa(a_i)$ for every $i = 1 \ldots n$.

But how to generalize this for choice rules $C \leftarrow C_1, \ldots, C_n$?

For every $a \in x$ there is some $r = C \leftarrow C_1, \ldots, C_n \in \mathcal{P}$ s.t. $a \in \mathrm{dom}(C)$ and

- d-grounded: $\kappa(a) > \max\{\kappa(b) \mid b \in \bigcup_{i=1}^{n} \mathrm{dom}(C_i)\}$,
- s-grounded: there is some x_i s.t. for every $z \in [x_i, x]$, $\mathrm{dom}(C_i) \cap z \in \mathrm{sat}(C_i)$ for every $i = 1 \ldots n$ and $\kappa(a) > max\{\kappa(b) \mid b \in x_i\}$.
- a-grounded: for every $i = 1 \ldots n$, there is some $z \subseteq \{b \mid \kappa(b) < \kappa(a)\}$ s.t. $z \cap \mathrm{dom}(C_i) \in \mathrm{sat}(C_i)$.

Groundedness: Results and Intuitiveness

Proposition

If x is d-grounded for $\mathcal{P}$, it is s-grounded for $\mathcal{P}$.

If x is s-grounded for $\mathcal{P}$, it is a-grounded for $\mathcal{P}$.

Theorem

Let $\mathcal{P}$ a normal logic program $\mathcal{P}$. Then x is a-grounded for $\mathcal{P}$ then x is grounded according to [Erdem and Lifschitz, 2003].

Operator	d-ground.	s-ground.	a-ground.
GZ	$\vee$	$\vee$	$\vee$
LPST	$\times$	$\vee$	$\vee$
MR	$\times$	$\times$	$\vee$
$\mathcal{U}$	$\times$	$\times$	$\times$
Counter-Example	$\{b \leftarrow 1\{a, b\}; a \leftarrow\}$	$\{a \leftarrow \{a, b\} \neq 1; b \leftarrow \{a, b\} \neq 1\}$	$\{\{p, q\} = 2 \leftarrow \{p, q\} \neq 1\}$
Dislikes	[Alviano et al., 2023]	[Alviano and Faber, 2019]	[Liu et al., 2010]

Disjunctions are Choice Constructs

Disjunctions are Choice Constructs

$\mathtt{D2C}(\mathcal{P}) = \{1\Delta \leftarrow \phi \mid \bigvee \Delta \leftarrow \phi \in \mathcal{P}\}$.

Example $\mathtt{D2C}(\{p \vee q \leftarrow .\}) = \{1\{p \vee q\} \leftarrow .\}$.

Disjunctions are Choice Constructs

$\mathtt{D2C}(\mathcal{P}) = \{1\Delta \leftarrow \phi \mid \bigvee \Delta \leftarrow \phi \in \mathcal{P}\}$.

Example $\mathtt{D2C}(\{p \vee q \leftarrow .\}) = \{1\{p \vee q\} \leftarrow .\}$.

Proposition

For any disjunctive logic program $\mathcal{P}$, $\mathcal{IC}_{\mathcal{P}} = \mathcal{IC}^{c}_{\mathtt{D2C}(\mathcal{P})}$.

Disjunctions are Choice Constructs

$\texttt{D2C}(\mathcal{P}) = \{1\Delta \leftarrow \phi \mid \bigvee \Delta \leftarrow \phi \in \mathcal{P}\}$.

Example $\texttt{D2C}(\{p \vee q \leftarrow .\}) = \{1\{p \vee q\} \leftarrow .\}$.

Proposition

For any disjunctive logic program $\mathcal{P}$, $\mathcal{IC}_{\mathcal{P}} = \mathcal{IC}^{c}_{\texttt{D2C}(\mathcal{P})}$.

Difference between DLP and choice programs: the difference is *not* in the treatment of disjunction and choice atoms (i.e. when they should be made true or false), but rather in how the stable semantics is constructed:

- Disjunctions: minimality-based stable semantics [**H**. et al., 2024]
  $S(\mathcal{O}_u)(x) \;=\; \{y \in \mathcal{L} \mid y \in \mathcal{O}_u(x, y) \text{ and } \texttt{not}\ \exists y' < y : y' \in \mathcal{O}_u(x, y')\}$
  $\{p\}$ and $\{q\}$ are stable
- Choice constructs: constructive stable semantics
  $\{p\}$, $\{p, q\}$ and $\{q\}$ are stable

Conclusion

- Operator-based study of choice constructs that gives us:
  - Whole family of operators and semantics (three-valued supported and stable, KK- and WF-state semantics, semi-equilibrium semantics) that generalise existing semantics.
  - Clear view on difference between disjunction and choice rules.
  - Allows to apply concepts from AFT (splitting, groundedness, conditional independence).
- Evaluation of stable semantics based on different operators according to various notions of groundedness.
- Future work: complexity, existence, implementation.

Conclusion

- Operator-based study of choice constructs that gives us:
  - Whole family of operators and semantics (three-valued supported and stable, KK- and WF-state semantics, semi-equilibrium semantics) that generalise existing semantics.
  - Clear view on difference between disjunction and choice rules.
  - Allows to apply concepts from AFT (splitting, groundedness, conditional independence).
- Evaluation of stable semantics based on different operators according to various notions of groundedness.
- Future work: complexity, existence, implementation.

Thank you for your attention. Questions?

Alviano, M. and Faber, W. (2019).
Chain answer sets for logic programs with generalized atoms.
In Calimeri, F., Leone, N., and Manna, M., editors, *Logics in Artificial Intelligence - 16th European Conference, JELIA2019, Rende, Italy, May 7-11, 2019, Proceedings*, volume 11468 of *Lecture Notes in Computer Science*, pages 462–478. Springer.

Alviano, M., Faber, W., and Gebser, M. (2023).
Aggregate semantics for propositional answer set programs.
Theory and Practice of Logic Programming, 23(1):157–194.

Erdem, E. and Lifschitz, V. (2003).
Tight logic programs.
Theory and Practice of Logic Programming, 3(4-5):499–518.

Gelfond, M. and Zhang, Y. (2014).
Vicious circle principle and logic programs with aggregates.
Theory and Practice of Logic Programming, 14(4-5):587–601.

H., J., Arieli, O., and Bogaerts, B. (2024).
Non-deterministic approximation fixpoint theory and its application in disjunctive logic programming.
Artif. Intell., 331:104110.

Bibliography iii

Liu, L., Pontelli, E., Son, T. C., and Truszczyński, M. (2010).
Logic programs with abstract constraint atoms: The role of computations.
Artificial Intelligence, 174(3-4):295–315.

Marek, V. W. and Remmel, J. B. (2004).
Set constraints in logic programming.
In *Logic Programming and Nonmonotonic Reasoning: 7th International Conference, LPNMR 2004 Fort Lauderdale, FL, USA, January 6-8, 2004 Proceedings 7*, pages 167–179. Springer.