ON LOGIC PROGRAM UPDATES

INVITED TALK AT EPCL 2012 WORKSHOP

João Leite
CENTRIA, New University of Lisbon
Non-Monotonic Logic Programming

- **Stable Models Semantics**: A semantics for logic programs with negation developed by M. Gelfond and V. Lifschitz (1987-91), which lead to:

  Answer-Set Programming (ASP)

- ASP has good properties for Knowledge Representation and Problem Solving
  - expressive language;
  - 0, 1 or multiple answer sets (models);
  - two forms of negation to reason with a limited combination of the closed and open world assumptions;
    - we restrict to default negation.
  - fast answer-set solvers (DLV, CLASP, SMODELS, etc...);
  - theoretically well understood language;
Logic Programs

- A (generalized) rule $r$ is:
  \[ L_0 \leftarrow L_1, \ldots, L_n. \]
  - where each $L_i$ is a literal i.e. an atom $A$ or default literal $\neg A$.
- $H(r) = L_0$ is the head of rule $r$
- $B(r) = \{L_1, \ldots, L_n\}$ is the body of rule $r$

- A (generalised) logic program is a set of rules

- Example:
  \[
  \begin{align*}
  a & \leftarrow. \\
  \neg a & \leftarrow b. \\
  b & \leftarrow \neg c. \\
  d & \leftarrow \neg e. \\
  c & \leftarrow \neg b.
  \end{align*}
  \]
Why default negation in heads?

- We need a way to update the truth value of an atom to “not being true”.
  - In a dynamic setting, updating with a rule
    \[ A \leftarrow L_1, \ldots, L_n. \]
    means that if \( L_1, \ldots, L_n \) is true, then \( A \) should now be true
  - while updating with a rule
    \[ \neg A \leftarrow L_1, \ldots, L_n. \]
    means that if \( L_1, \ldots, L_n \) is true, then \( A \) should now not be true

- Why not use strong (classical) negation in the head instead?
  - LPs with two kinds of negation allow three different (consistent) states wrt. some atom \( A \), namely \( \{A\} \), \( \{\neg A\} \) and \( \{\} \).
  - We need to be able to update from/to any of these states
    - Strong negation updates to \( \{\neg A\} \)
    - Default negation updates to \( \{\} \)
Logic Programs

An interpretation \( I \) is a stable model of a program \( P \) if:

\[
I' = \text{least}(P \cup \text{Defaults})
\]

- \( I' = I \cup \{\neg A \mid A \text{ is an atom and } A \not\in I\} \)
- \( \text{Defaults} = \{\neg A \mid A \text{ is an atom and } A \not\in I\} \)
- \( \text{least(.)} \) denotes the least model of the (positive) program obtained by treating literals of the form \( \neg A \) as new atoms.

Example:

\( P = \{a. \quad b \leftarrow \neg c. \quad c \leftarrow \neg b. \quad \neg a \leftarrow b. \quad d \leftarrow \neg e.\} \)
\( I = \{a, c, d\} \quad I' = \{a, \neg b, c, d, \neg e\} \quad \text{Defaults} = \{\neg b, \neg e\} \)

\[
\text{least}(P \cup \text{Defaults}) = \\
= \text{least}(\{a. b \leftarrow \neg c. c \leftarrow \neg b. \neg a \leftarrow b. d \leftarrow \neg e.\} \cup \{\neg b. \neg e.\}) = \\
= \{a, \neg b, c, d, \neg e\} = I' \\
\Rightarrow \{a, c, d\} \text{ is a stable model.}
\]
Belief Change

- Change operations on monotonic logics have been studied extensively in the area of belief change.
  - rationality postulates for operations play a central role
  - constructive operator definitions correspond to sets of postulates
- two different belief change operations have been distinguished [Katsuno and Mendelzon1991]:
  - **Revision**
    - recording newly acquired information about a static world
    - characterized by AGM postulates and their descendants
  - **Update**
    - recording changes in a dynamic world
    - characterized by KM postulates for update
KM Postulates

[Katsuno and Mendelzon 91]

Postulates (KM 1) – (KM 8)

(KM 1) \( \phi \circ \psi \models \psi \).

(KM 2) If \( \phi \models \psi \), then \( \phi \circ \psi \equiv \phi \).

(KM 3) If both \( \phi \) and \( \psi \) are satisfiable, then \( \phi \circ \psi \) is satisfiable.

(KM 4) If \( \phi_1 \equiv \phi_2 \) and \( \psi_1 \equiv \psi_2 \), then \( \phi_1 \circ \psi_1 \equiv \phi_2 \circ \psi_2 \).

(KM 5) \( (\phi \circ \psi) \land \chi \models \phi \circ (\psi \land \chi) \).

(KM 6) If \( \phi \circ \psi_1 \models \psi_2 \) and \( \phi \circ \psi_2 \models \psi_1 \), then \( \phi \circ \psi_1 \equiv \phi \circ \psi_2 \).

(KM 7) \( (\phi \circ \psi_1) \land (\phi \circ \psi_2) \models \phi \circ (\psi_1 \lor \psi_2) \) if \( \phi \) is complete.

(KM 8) \( (\phi_1 \lor \phi_2) \circ \psi \equiv (\phi_1 \circ \psi) \lor (\phi_2 \circ \psi) \).
Logic Program Updates

- Problem
  - Assigning Semantics to a sequence of Logic Programs: $(P_1, P_2, \ldots, P_n)$

- ...or a Dynamic Logic Program

- Several lines of research
  - Based on Causal Rejection
  - Based on Abduction/Priorities/Preferences
  - Based on KM Postulates
  - Based on Structural Properties
  - ...
Fact Updates

When the initial knowledge is just a set of facts \( (I_i) \)
- an interpretation \( I_u \) is a Justified Update of \( I_i \) by a program \( Q \) if
  - \( I_u \) is a model of \( Q \)
  - There is no other model \( I_x \) of \( Q \) such that \( \Delta(I_x,I_i) \subseteq \Delta(I_u,I_i) \)

Example:
\[ I_i = \{\text{rain,clowdy}\} \]
\[ Q = \sim \text{rain} \leftarrow \text{play} \leftarrow \sim \text{rain} \]
\[ I_u = \{\text{play,clowdy}\} \]

If the initial program is just a set of facts, then the result of updating it should be like in Fact Updates.

**\( P_{IU} \): Generalisation of Fact Updates**
\[ P_i = \{A \leftarrow | A \in I\} \Rightarrow \text{SEM}(P_i \oplus Q) = IU(I,Q) \]
Program Updates

- What if our initial KB is a Logic Program?
- Can we simply take each of its stable models and update it?

- Initial Program P:
  
  sleep ← ~tv_on.
  tv_on.
  watch_tv ← tv_on.

- Stable Model:
  
  \{tv_on, watch_tv\}

- Intended Model is \{power_failure, sleep\}!

- Update Program U:
  
  power_failure.
  ~tv_on ← power_failure.

- Updated Model:
  
  \{power_failure, watch_tv\}
Truth value of any element should be supported by some rule (either from the update program or from the initial program).

\[ P_\sigma: \text{Support:} \]

\[
\text{if } a \in M \text{ then } \exists r \in P_i, H(r) = a \land M \models B(r)
\]

Inertia should be exerted on the program rules instead of model literals.

Inertia in rules should only be blocked (or rules rejected) if there is a newer directly conflicting rule (or cause).

\[ P_\gamma: \text{Causal Rejection:} \]

\[
\text{if } M \not\models r \in P_j \text{ then } \exists r' \in P_k, j < k, H(r) = \neg H(r') \land M \models B(r')
\]
Other Desirable Properties

\( P_\nu : \) Primacy of new information
\[ M \in \text{SEM}(P \oplus Q) \Rightarrow M \models Q \]

\( P_\emptyset : \) Immunity to empty updates
\[ \text{SEM}(P \oplus \emptyset) = \text{SEM}(\emptyset \oplus P) = \text{SEM}(P) \]

\( P_\tau : \) Immunity to tautologies
\[ \text{SEM}(P \oplus Q) = \text{SEM}(P \oplus (Q \cup \{\tau\})) = \text{SEM}((P \cup \{\tau\}) \oplus Q) \]
where \( \tau \) is any tautology i.e. any rule \( \tau \) such that \( H(\tau) \in B(\tau) \)

\( P_{\rho\varepsilon} : \) Refined Extension Principle
Generalisation of \( P_\tau \) to certain circular updates.
An interpretation $I$ is a stable model of $(P_1, \ldots, P_n)$ if

$$I' = \text{least}( \bigcup (P_i) - \text{Reject}(I)) \cup \text{Defaults}(I) )$$
DLP – Justified Updates

Interpretation I is a Justified Update of \((P_1, \ldots, P_n)\) if

\[
I' = \text{least}( \bigcup(P_i) - \text{Reject}(I) ) \cup \text{Defaults}(I)
\]

\[
\text{Reject}(I) = \{ r \in P_i | \exists r' \in P_j, i < j, \ H(r) = \neg H(r') \land I \models B(r') \}
\]

\[
\text{Defaults}(I) = \{ \neg A | A \text{ is an atom and } A \notin I \}
\]

[\text{L and Pereira 98}]

\[14\]

© J. Leite, CENTRIA

On Logic Program Updates, Invited Talk at EPCL’12 Workshop
DLP – Justified Updates

Interpretation $I$ is a Justified Update of $(P_1,\ldots,P_n)$ if

$$I' = \text{least} \left( \bigcup (P_i) - \text{Reject}(I) \right) \cup \text{Defaults}(I)$$

$$\text{Reject}(I) = \{ r \in P_i \mid \exists r' \in P_j, i < j, H(r) = \neg H(r') \land I \models B(r') \}$$

$$\text{Defaults}(I) = I^-$$

Initial Program $P_1$:
- sleep ← ~tv_on.
- tv_on.
- watch_tv ← tv_on.

Update Program $P_2$:
- power_failure.
- ~tv_on ← power_failure.

Intended Model $I = \{ \text{sleep, power_failure} \}$

$I' = \{ \text{sleep, power_failure, ~tv_on, ~watch_tv} \}$

$\text{Reject}(I) = \{ \text{tv_on} \}$

$\text{Defaults}(I) = \{ \text{~tv_on, ~watch_tv} \}$

least\{sleep ← ~tv_on. watch_tv ← tv_on.
- power_failure. ~tv_on ← power_failure. ~tv_on. ~watch_tv.\} =

$= \{ \text{sleep, power_failure, ~tv_on, ~watch_tv} \} = I'$
Properties:

- **P∅**: Immunity to empty updates
  \[ \text{SEM}(P \oplus \emptyset) = \text{SEM}(\emptyset \oplus P) = \text{SEM}(P) \]

- **Pν**: Primacy of new information
  \[ M \in \text{SEM}(P \oplus Q) \Rightarrow M \vdash Q \]

- **Pσ**: Support
  \[ A \in M \in \text{SEM}(P \oplus Q) \Rightarrow \exists r \in (P \cup Q) : H(r) = A \land M \vdash B(r) \]
But, it doesn’t obey:

\( P_\tau : \text{Immunity to tautologies} \)

\[
\text{SEM}(P \oplus Q) = \text{SEM}(P \oplus (Q \cup \{\tau\})) = \text{SEM}((P \cup \{\tau\}) \oplus Q)
\]

where \( \tau \) is any tautology i.e. any rule \( \tau \) such that \( H(\tau) \in B(\tau) \)
But, it doesn’t obey:

\[ P_\tau : \text{Immunity to tautologies} \]

\[
\text{SEM}(P \oplus Q) = \text{SEM}(P \oplus (Q \cup \{\tau\})) = \text{SEM}((P \cup \{\tau\}) \oplus Q)
\]

where \( \tau \) is any tautology, i.e., any rule \( \tau \) such that \( H(\tau) \in B(\tau) \).

---

**Justified Updates and Immunity to Tautologies**

<table>
<thead>
<tr>
<th>( P_1 )</th>
<th>( P_2 )</th>
<th>Intended</th>
<th>Obtained</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a. )</td>
<td>( \sim a \leftrightarrow \sim a. )</td>
<td>{( a }}</td>
<td>{( a }} and {}</td>
</tr>
<tr>
<td>( \sim a. )</td>
<td>( a \leftrightarrow a. )</td>
<td>{( a }}</td>
<td>{}</td>
</tr>
</tbody>
</table>

Intended -no models- Obtained

\[ \times \times \]
And, it also doesn’t obey:

\[ P_{\uparrow} : \text{Generalisation of Fact Updates} \]

\[ P_{\uparrow} = \{ A \leftarrow | A \in I \} \Rightarrow \text{SEM}(P_{\uparrow} \oplus Q) = IU(I, Q) \]
And, it also doesn’t obey:

$P_w: \text{Generalisation of Fact Updates}$

$\overline{P_1} = \{A \leftarrow | A \in I\} \Rightarrow \text{SEM}(P_1 \oplus Q) = IU(I, Q)$
Interpretation $I$ is a Dynamic Stable Model of $(P_1, \ldots, P_n)$ if

$$I' = \text{least}( \bigcup (P_i) - \text{Reject}(I) ) \cup \text{Defaults}(I) \)$$

$$\text{Reject}(I) = \{ r \in P_i | \exists r' \in P_j, i < j, H(r) = \neg H(r') \land I \models B(r') \}$$

$$\text{Defaults}(I) = \{ \neg A | \nexists r \in P_i, H(r) = A \land I \models B(r) \}$$
DLP — Dynamic Stable Models

[Alferes, L, Pereira, Przymusinska and Przymusinski 98,00]

□ Properties:

\( P_\emptyset \): Immunity to empty updates

\[ \text{SEM}(P \oplus \emptyset) = \text{SEM}(\emptyset \oplus P) = \text{SEM}(P) \]

\( P_\nu \): Primacy of new information

\[ M \in \text{SEM}(P \oplus Q) \Rightarrow M \vDash Q \]

\( P_\sigma \): Support

\[ A \in M \in \text{SEM}(P \oplus Q) \Rightarrow \exists r \in (P \cup Q) : H(r) = A \land M \vDash B(r) \]

\( P_\iota \): Generalisation of Fact Updates

\[ P_\iota = \{ A \leftarrow \mid A \in I \} \Rightarrow \text{SEM}(P_\iota \oplus Q) = IU(I,Q) \]
DLP – Dynamic Stable Models

[Alferes, L, Pereira, Przymusinska and Przymusinski 98,00]

But, it doesn’t obey:

$P_\tau$: Immunity to tautologies

$$\text{SEM}(P \oplus Q) = \text{SEM}(P \oplus (Q \cup \{\tau\})) = \text{SEM}((P \cup \{\tau\}) \oplus Q)$$

where $\tau$ is any tautology i.e. any rule $\tau$ such that $H(\tau) \in B(\tau)$
But, it doesn’t obey:

\( P_\tau : \text{Immunity to tautologies} \)

\[
\text{SEM}(P \oplus Q) = \text{SEM}(P \oplus (Q \cup \{\tau\})) = \text{SEM}((P \cup \{\tau\}) \oplus Q)
\]

where \( \tau \) is any tautology, i.e., any rule \( \tau \) such that \( H(\tau) \in B(\tau) \).

**Dynamic Stable Models and Immunity to Tautologies**

<table>
<thead>
<tr>
<th>P₁: ( a )</th>
<th>P₂: ( \sim a \leftarrow \sim a )</th>
<th>Intended ( {a} )</th>
<th>Obtained ( {a} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>P₁: ( a )</td>
<td>P₂: ( \sim a \leftarrow \sim a )</td>
<td>Intended ( {a} )</td>
<td>Obtained ( {a} )</td>
</tr>
<tr>
<td>P₁: ( a )</td>
<td>P₂: ( a \leftarrow a )</td>
<td>Intended ( {a} )</td>
<td>Obtained ( {a} )</td>
</tr>
</tbody>
</table>
And it also doesn’t obey:

\( P_{\rho_\varepsilon} \): Refined Extension Principle

Generalisation of \( P_\tau \) to certain circular updates.
And it also doesn't obey:

\[ P_{\rho e} : \text{Refined Extension Principle} \]

**Generalisation of** \( P_\tau \) **to certain circular updates.**

[Alferes, L, Pereira, Przymusinska and Przymusinski 98,00]

### Dynamic Stable Models and Refined Extension Principle

| P₁: a ← ~b.  | Intended: \{a\}  | Obtained: \{a\} and \{b,c\} | \( \times \) |
| b ← ~a.       |               |                                 |         |
| c ← b.        |               |                                 |         |
| ~c.           |               |                                 |         |
| P₂: c ← c.    |               |                                 |         |

| P₁: a ← ~b.  | Intended: \{a\}  | Obtained: \{a\} and \{b,c,e\} | \( \times \) |
| b ← ~a.       |               |                                 |         |
| c ← b.        |               |                                 |         |
| ~c.           |               |                                 |         |
| P₂: c ← e.    |               |                                 |         |
| e ← c.        |               |                                 |         |
DLP – Refined Dynamic Stable Models

[Alferes, Banti, Brogi and L 05]

- Interpretation I is a Refined Dynamic Stable Model of \((P_1, \ldots, P_n)\) if

\[
I' = \text{least}( \bigcup(P_i) - \text{Reject}(I)) \cup \text{Defaults}(I)
\]

\[
\text{Reject}(I) = \{ r \in P_i | \exists r' \in P_j, i \leq j, H(r) = \neg H(r') \land I \models B(r') \}
\]

\[
\text{Defaults}(I) = \{ \neg A | \not\exists r \in P_i, H(r) = A \land I \models B(r) \}
\]
Interpretation $I$ is a Refined Dynamic Stable Model of $(P_1, \ldots, P_n)$ if

$$I' = \text{least}( \bigcup (P_i) - \text{Reject}(I) ) \cup \text{Defaults}(I) )$$

- **Reject**$(I) = \{ r \in P_i | \exists r' \in P_j \land i \leq j, H(r) = \neg H(r') \land I \models B(r') \}$
- **Defaults**$(I) = \{ \neg A | \not\exists r \in P_i \land H(r) = A \land I \models B(r) \}$

**Initial Program** $P_1$:
- $a$
- $\neg a$

**Update Program** $P_2$:
- $a \leftarrow a$

Unintended Model $I = \{ a \}$

Reject$(I) = \{ a, \neg a \}$

Defaults$(I) = \{ \}$

least$\{ a \leftarrow a \} = \{ \} \neq I' = \{ a \}$
DLP – Refined Dynamic Stable Models

[Alferes, Banti, Brogi and L 05]

- Properties:

  \( P_{\emptyset} \): Immunity to empty updates

  \[
  \text{SEM}(P \oplus \emptyset) = \text{SEM}(\emptyset \oplus P) = \text{SEM}(P)
  \]

  \( P_{\tau} \): Immunity to tautologies

  \[
  \text{SEM}(P \oplus Q) = \text{SEM}(P \oplus (Q \cup \{\tau\})) = \text{SEM}((P \cup \{\tau\}) \oplus Q)
  \]

  where \( \tau \) is any tautology i.e. any rule \( \tau \) such that \( H(\tau) \in B(\tau) \)

  \( P_{\nu} \): Primacy of new information

  \[
  M \in \text{SEM}(P \oplus Q) \Rightarrow M \models Q
  \]

  \( P_{\sigma} \): Support

  \[
  A \in M \in \text{SEM}(P \oplus Q) \Rightarrow \exists r \in (P \cup Q) : H(r) = A \land M \models B(r)
  \]

  \( P_{\psi} \): Generalisation of Fact Updates

  \[
  P_{I} = \{A \leftarrow |A \in I\} \Rightarrow \text{SEM}(P_{I} \oplus Q) = IU(I,Q)
  \]

  \( P_{\rho e} \): Refined Extension Principle

  Generalisation of \( P_{\tau} \) to certain circular updates.
DLP – Dynamic Answer Sets

[Unpublished work by Eiter, Fink, Sabbatini and Tompits]

- Interpretation \( I \) is a Dynamic Answer Set of \((P_1, \ldots, P_n)\)
  if

\[
I' = \text{least}( \bigcup (P_i) - \text{Reject}(I)) \cup \text{Defaults}(I)
\]

\[
\text{Reject}(I) = \{ r \in P_i | \exists r' \in P_j \setminus \text{Reject}(I), i < j, H(r) = \neg H(r') \land I \models B(r') \}
\]

 Defaults(I) = I'
DLP – Dynamic Answer Sets

[ Eiter, Fink, Sabbatini and Tompits 02 ]

□ It doesn’t obey:

\( P_\tau : \text{Immunity to tautologies} \)

\[
\text{SEM}(P \oplus Q) = \text{SEM}(P \oplus (Q \cup \{\tau\})) = \text{SEM}((P \cup \{\tau\}) \oplus Q)
\]

where \( \tau \) is any tautology i.e. any rule \( \tau \) such that \( H(\tau) \in B(\tau) \)
It doesn’t obey:

\[ P_\tau : \text{Immunity to tautologies} \]

\[ \text{SEM}(P \oplus Q) = \text{SEM}(P \oplus (Q \cup \{\tau\})) = \text{SEM}((P \cup \{\tau\}) \oplus Q) \]

where \( \tau \) is any tautology, i.e., any rule \( \tau \) such that \( H(\tau) \in B(\tau) \).
Relationship between Semantics

Dynamic Answer Sets → Justified Updates → Dynamic Stable Models → Refined Dynamic Stable Models

They all coincide for acyclic LPs [Homola04, Banti et al. 05]
### Summary of Properties

<table>
<thead>
<tr>
<th></th>
<th>$P_\emptyset$</th>
<th>$P_\tau$</th>
<th>$P_\nu$</th>
<th>$P_\sigma$</th>
<th>$P_\psi$</th>
<th>$P_\rho\varepsilon$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Immunity to empty updates</td>
<td>Immunity to tautologies</td>
<td>Primacy of new information</td>
<td>Support</td>
<td>Generalisation of Fact Updates</td>
<td>Refined Extension Principle</td>
</tr>
<tr>
<td>Justified Updates</td>
<td>✔️</td>
<td>✗</td>
<td>✔️</td>
<td>✔️</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Dynamic Stable Models</td>
<td>✔️</td>
<td>✗</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✗</td>
</tr>
<tr>
<td>Dynamic Answer Sets</td>
<td>✔️</td>
<td>✗</td>
<td>✔️</td>
<td>✔️</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Refined Dynamic Stable Models</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>
Other Approaches

- Preference-based Semantics
  - Program Updates through Priorities
    - Zhang 06
  - Program Updates through Preferences
    - Delgrande et al. 07.
  - Revision Semantics
    - Delgrande 10

- Abduction-based Semantics
  - Sakama and Inoue 03

- Using structural properties
  - Krumpelmann and Kern-Isberner 10
  - Sefranek 06
Updates through a complex mixture of:
- Fact Updates
- Logic Programs with Priorities.

To determine $P \oplus Q$:
- For each Stable Model $M$ of $P$, determine $M' = IU(M,Q)$
- Determine a maximal subset of $P$, $P'$, coherent with $M'$.
- Define a prioritised Logic Program $(P', Q)$ with $Q > P'$
- Finally, determine the reducts of $(P', Q)$ which are the result of updating $P$ with $Q$. 

[Zhang 06]
Updates through a complex mixture of:

- Fact Updates
- Logic Programs with Priorities.

To determine $P \oplus Q$:

- For each Stable Model $M$ of $P$, determine $M' = IU(M, Q)$
- Determine a maximal subset of $P$, $P'$, coherent with $M'$.
- Define a prioritised Logic Program $(P', Q)$ with $Q > P'$
- Finally, determine the reducts of $(P', Q)$ which are the result of updating $P$ with $Q$. 

[Zhang 06]
Updates through a complex mixture of:

- Fact Updates
- Logic Programs with Priorities

To determine $P \oplus Q$:

1. For each Stable Model $M$ of $P$, determine $M' = IU(M,Q)$.
2. Determine a maximal subset of $P$, $P'$, coherent with $M'$.
3. Define a prioritised Logic Program $(P', Q)$ with $Q > P'$.
4. Finally, determine the reducts of $(P', Q)$ which are the result of updating $P$ with $Q$. 

Program Updates Through Priorities [Zhang06] and Immunity to Tautologies

\[ \begin{align*} 
P_1: & \quad a \leftarrow \neg \neg a. \\
     & \quad \neg a \leftarrow \neg a. \\
P_2: & \quad a \leftarrow a. \\
\end{align*} \]

Intended \{a\} and \{\neg a\}

Obtained \{a\}
Program Updates Through Priorities

Updates through a complex mixture of:
- Fact Updates
- Logic Programs

To determine $P^\lor Q$:
- For each Stable Model $M$ of $P$, determine $M'=IU(M,Q)$
- Determine a maximal subset of $P$, $P'$, coherent with $M'$.
- Define a prioritised Logic Program $(P',Q)$ with $Q>P'$
- Finally, determine the reducts of $(P',Q)$ which are the result of updating $P$ with $Q$.

Program Updates Through Priorities [Zhang06] and Undetected Conflicts

| $P_1$: $a \leftarrow c$, $b \leftarrow c$ | Intended $\{\neg a, b, c\}$ | Obtained -no models- $\times$ |
Program Updates Through Preferences
[Delgrande, Schaub and Tompits 07]

- Updates through a mixture of:
  - Preferences
  - Defeasible Rules

- The update models of \( P \oplus Q \) are the preferred models of a prioritised Logic Program \((\overline{\Pi},<)\) constructed in one of three possible ways (three different operators):
  - \((P^d \cup Q^d, P^d \times Q^d)\)
  - \((P^d \cup Q^d, C(P^d, Q^d))\)
  - \((c(PUQ)^d \cup ((PUQ) \setminus c(PUQ)), C(P^d, Q^d))\)

where
- \(P^d\) stands for the defeasible version of \(P\) (i.e. obtained from adding \(\sim \neg \text{head}(r)\) to the body of every rule \(r\)).
- \(C(P \times Q)\) stands for pairs of rules with conflicting heads
- \(c(PUQ)\) stands for the rules in \(C(P \times Q)\).
Updates through a mixture of:
- Preferences
- Defeasible Rules

The update models of $P \oplus Q$ are the preferred models of a prioritised Logic Program $(\Pi, \prec)$ constructed in one of the three possible ways (that are denoted as $\sigma$): 
- $(P^d \cup Q^d, P^d \times Q^d)$
- $(P^d \cup Q^d, C(P^d, Q^d))$
- $(c(P \cup Q)^d \cup (P \cup Q) \setminus C(P \cup Q), C(P^d, Q^d))$

where $P^d$ stands for the defeasible version of $P$ (i.e. obtained from adding $\neg \neg \text{head}(r)$ to the body of every rule $r$).
- $C(P \times Q)$ stands for pairs of rules with conflicting heads
- $c(P \cup Q)$ stands for the rules in $C(P \times Q)$.

Updates Through Preferences [Delgrande et al. 07] and Default Assumptions

- $P_1: \neg a.$
- $P_2: a \leftarrow \neg \neg a.$

Obtained $\{a\}$
Updates are determined by:

- Taking the most recent program and commit to a maximal set of default assumptions (default literals) needed to build one of its answer-sets.
- Then, add a maximal coherent sub-set of rules of the predecessor program, and commit to more default assumptions.
- ...
Revision Semantics

Updates are determined by:

- Taking the most recent program and commit to a maximal set of default assumptions (default literals) needed to build one of its answer-sets.

Then, add a maximal coherent subset of rules of the predecessor program, and commit to more default assumptions.

---

Revision Semantics [Delgrande 10] and Default Assumptions

- $P_1$: $a$
- $P_2$: $b \leftarrow \neg a$

Intended: \{a\}

Obtained: \{b\} (✓)

...
Program Updates Through Abduction

[ Sakama and Inoue 03 ]

- Updates through Abduction:
  - $P' = (P \cup Q) \setminus R$ is the result of $P \oplus Q$ if:
    - $\text{SEM}(P') \neq \emptyset$
    - $R \subseteq P$
    - $\not\exists R' \subset R | \text{SEM}((P \cup Q) \setminus R') \neq \emptyset$

- Main Problem – fails even the most basic property
  - $P \emptyset$: Immunity to empty updates
    - $\text{SEM}(P \oplus \emptyset) = \text{SEM} (\emptyset \oplus P) = \text{SEM}(P)$

- Other issues:
  - Commits to rejected rules ($R$), which cannot be reused.
  - The result can be more than one program.
  - Higher Computational Complexity.
Other Properties

\( P_{\mu\rho} : \) Minimal Rule Rejection

\[ \text{SEM}(P \cup Q) \neq \emptyset \Rightarrow \text{SEM}(P \oplus Q) = \text{SEM}(P \cup Q) \]

\( P_{\omega\mu\chi} : \) Weak Minimal Change

\[ \text{SEM}(P \cup Q) \neq \emptyset \Rightarrow \text{SEM}(P \oplus Q) \subseteq \text{SEM}(P \cup Q) \]

\( P_{\nu\rho} : \) Universal Recoverability Principle

\[ \forall P \exists Q : \text{SEM}(P \oplus Q) \neq \emptyset \]
## Summary of Properties

<table>
<thead>
<tr>
<th></th>
<th>$P_\emptyset$</th>
<th>$P_\tau$</th>
<th>$P_\nu$</th>
<th>$P_\sigma$</th>
<th>$P_\omega$</th>
<th>$P_\rho$</th>
<th>$P_\mu$</th>
<th>$P_\omega_\chi$</th>
<th>$P_\nu_\rho$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Justified Updates</strong></td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Dynamic Stable Models</strong></td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Dynamic Answer Sets</strong></td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Refined Dynamic Stable Models</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>LP Updates through Abduction</strong></td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>LP Updates through Priorities</strong></td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>LP Updates through Preferences$^1,2$</strong></td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>-</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td><strong>LP Updates through Preferences$^3$</strong></td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>-</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td><strong>Revision Semantics</strong></td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>-</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

---

© J. Leite, CENTRIA
What about Classical Belief Change?

- Directly applying the KM postulates and constructions from belief change to logic programs and answer-sets leads to a number of serious problems.
  - ambiguity of the postulates, often difficult to formulate for logic programs and answer-sets
  - leads to very counterintuitive results
  - at the heart of [Leite and Pereira 98] and thoroughly investigated in [Eiter, Fink, Sabbatini and Tompits 02]

- Reconciliation of belief change with rule evolution is still a very interesting open problem:
  - a more general understanding of knowledge evolution
  - a semantic approach to rule evolution, focusing only on the meaning of a logic program and not on its syntactic representation

- How to proceed?
Belief Change and SE Models

- Belief Change on SE Models
  - AGM Revision on SE Models
    - [Delgrande, Schaub, Tompits and Woltran 08]
- SE Models [Turner 03]
  - semantic characterisation of logic programs, coinciding with the models in the Logic of Here and There for the fragment corresponding to logic programs.
  - richer structure – an SE interpretation $X$ is a pair of ordinary interpretations $<I,J>$ such that $I \subseteq J$
  - an interpretation $<I,J>$ is an SE-model of a program $P$ if $J$ is a model of $P$ and $I$ is a model of $P^J$ (the GL reduct of $P$ by $I$)
  - monotonic and more expressive than answer sets
  - characterise strong equivalence
Postulates (PU 1) – (PU 8)

(PU 1) $P \oplus Q \models_s Q$.

(PU 2) If $P \models_s Q$, then $P \oplus Q \equiv_s P$.

(PU 3) If both $P$ and $Q$ are satisfiable, then $P \oplus Q$ is satisfiable.

(PU 4) If $P_1 \equiv_s P_2$ and $Q_1 \equiv_s Q_2$, then $P_1 \oplus Q_1 \equiv_s P_2 \oplus Q_2$.

(PU 5) $(P \oplus Q) \land R \models_s P \oplus (Q \land R)$.

(PU 6) If $P \oplus Q_1 \models_s Q_2$ and $P \oplus Q_2 \models_s Q_1$, then $P \oplus Q_1 \equiv_s P \oplus Q_2$.

(PU 7) $(P \oplus Q_1) \land (P \oplus Q_2) \models_s P \oplus (Q_1 \lor Q_2)$ if $P$ is basic.

(PU 8) $(P_1 \lor P_2) \oplus Q \equiv_s (P_1 \oplus Q) \lor (P_2 \oplus Q)$. 
KM Updates and SE Models

- **Construction:**

  \( \omega \) - assigns a partial order \( \leq^X_\omega \) to every SE interpretation \( X \)

  \[
  \langle [P \oplus Q]^{SE} = \bigcup_{X \in [P]^{SE}} \min([Q]^{SE}, \leq^X_\omega) \quad (1)
  \]

- **Representation Theorem:** A program update operator \( \oplus \) satisfies conditions (PU 1) – (PU 8) if and only if there exists a faithful and organised SE partial order assignment \( \omega \) such that (1) is satisfied for all programs \( P;Q \).

- We also defined a concrete operator.

- Great!

- But…
Problem with SE Model Update

[Slota and L 10]

- **Theorem** A program update operator that satisfies (PU4) either does not respect support or it does not respect fact update.

- **Proof**
  - Let $\oplus$ be a program update operator that satisfies PU4 and let:
    
    | P1:   | p. |
    | P2:   | p←q. |
    | Q:    | \neg q. |
    | q.    | q. |
    
  - Since $P_1 \equiv_s P_2$, by (PU4) we have that $P_1 \oplus Q \equiv_s P_2 \oplus Q$. Consequently, $P_1 \oplus Q$ has the same answer sets as $P_2 \oplus Q$.
  - Since $\oplus$ respects fact update, then $P_1 \oplus Q$ has the unique answer set $\{p\}$.
  - But then $\{p\}$ is an answer set of $P_2 \oplus Q$ in which $p$ is unsupported by $P_2 \cup Q$.
  - Hence $\oplus$ does not respect support.
How to Proceed?

Three ways to proceed:

- abandon the classical postulates and constructions
- use existing approaches (with a syntactic flavour)
  - Refined Dynamic Stable Models
- find a more expressive characterisation of logic programs
How to Proceed?

- **Our idea:**
  - View a Program as the Set of Sets of SE models of the rules it is composed of.
  - \( P_1: \{r. s.\} \) viewed as \( \{\langle r, r\rangle, \langle r, rs\rangle, \langle rs, rs\rangle\}, \{\langle s, s\rangle, \langle s, rs\rangle, \langle rs, rs\rangle\}\} \)
  - \( P_2: \{r \leftarrow s. \} \) viewed as \( \{\langle \emptyset, \emptyset\rangle, \langle 0, r\rangle, \langle r, r\rangle, \langle 0, rs\rangle, \langle r, rs\rangle, \langle rs, rs\rangle\}, \{\langle s, s\rangle, \langle s, rs\rangle, \langle rs, rs\rangle\}\} \)

- **Closer to Base Change**

- **But…**

  - \( P_1: \neg a \leftarrow b. \quad P_2: \neg b \leftarrow a. \quad P_3: \leftarrow a, b. \)

  ...are all SE-Equivalent because their rules are not distinguishable by the SE-models semantics!

  ...and we want to distinguish their effect when used to update the program \( \{a. \quad b.\}\)
How to proceed?

- Three ways to proceed:
  - abandon the classical postulates and constructions
  - use existing approaches (with a syntactic flavour)
    - Refined Dynamic Stable Models
  - find a more expressive characterisation of logic programs …
    - …not based on the Logic of Here and There (... and SE-models)!
RE-Models

- An interpretation \(<I, J>\) is an RE-model of a program \(P\) if \(I\) is a model of \(P^J\).
- Distinguishes
  \[
  P_1: \neg a \leftarrow b. \quad P_2: \neg b \leftarrow a. \quad P_3: \leftarrow a, b.
  \]
- Viewing a program as the set of sets of RE-models of its rules
- ... we defined an update operator that coincides with Justified Updates (apart from programs with local cycles).
- It can be seen as a semantic counterpart of Justified Updates.
Conclusions/Open Problems

- Semantic counterpart of Refined Dynamic Stable Models.
- Other notions of equivalence, instead of the one based on RE-Models, that allow us to satisfy some additional KM postulates.
  - Difficulty resides in capturing non-tautological irrelevant updates [Alferes et al 05, Sefranek 06].
- Better understanding of differences between Revision and Update in Logic Programming.
- Postulates for Updates of LPs
  - Although we should proceed with caution...
Conclusion…

The journey isn’t over…

… but we are getting there.
References


References

References