Causal Graph Justifications of Stable Models

Jorge Fandinno

Joint work with:
Pedro Cabalar and Michael Fink

1- University of Corunna, SPAIN
2- Vienna University of Technology, AUSTRIA

Logical Reasoning and computation
Corunna, Spain

February 24th, 2015
Causality and Knowledge Representation

- For **Knowledge Representation**, not just deriving conclusions but sometimes we require **explanations**

- **Causality**: is a quite common concept in **human daily discourse**. Present in (chronologically or physically) **distant cultures**.

- What “**A has caused B**” actually means?
For **Knowledge Representation**, not just deriving conclusions but sometimes we require **explanations**.

**Causality**: is a quite common concept in **human daily discourse**. Present in (chronologically or physically) **distant cultures**.

What “**A has caused B**” actually means?

- Sufficient cause
- Necessary cause
- Actual or contributory cause
Joint interaction

Example

- There is a law asserts that *driving drunk* is *punishable*.
- Suppose that some person drove drunk.

Take the logic program consisting of one rule and two labelled facts:

\[\text{punish} \leftarrow \text{drive, drunk} \quad \text{d : drive} \quad \text{k : drunk}\]

- Joint interaction of multiple events.
  The cause formed by \{d, k\} together has caused *punish*.
Joint interaction

Example

- There is a law asserts that \textit{driving drunk} is \textit{punishable}.
- Suppose that some person drove drunk.

Take the logic program consisting of one rule and two labelled facts:

\begin{align*}
punish & \leftarrow drive, drunk \\
d & : drive \\
k & : drunk
\end{align*}

- Joint interaction of multiple events.
  The cause formed by \{d, k\} together has caused \textit{punish}.

- Two kinds of causal rules:
  - Unlabelled rules: tracing them is irrelevant for causal purposes.
  - Labelled rules: keep track of possible ways to derive an effect.
We may want to keep track of involved rules and not only facts:

**Example**

- Law \( \ell \) asserts that *driving drunk* is *punishable* with imprisonment.
- The execution \( e \) of a sentence establishes that people who are *punished* are *imprisoned*.
- Suppose that some person drove drunk.

\[
\ell : \text{punish} \leftarrow \text{drive, drunk} \quad \quad \quad d : \text{drive} \quad \quad \quad k : \text{drunk}
\]
\[
e : \text{prison} \leftarrow \text{punish}
\]

We get a cause in the form of a label graph

\[
\begin{array}{ccc}
d & \ell & k \\
\ell & & e
\end{array}
\]
Main ideas

- **Multi-valued** semantics for logic programs: each true atom will be associated to a set of justifications (*causal graphs*)

- Accordingly, **falsity** = lack of justification.
  - This coincides with the informal reading for default negation: 
    
    \[
    \text{not } p = \text{there is no way to derive } p
    \]
Main ideas

- **Multi-valued** semantics for logic programs: each true atom will be associated to a set of justifications (*causal graphs*).

- Accordingly, **falsity = lack of justification**.
  - This coincides with the informal reading for *default negation*: $\textit{not } p = \text{there is no way to derive } p$

- Causes must be **non-redundant**.
  - Some causes will be **stronger** than others.
  - This allows us defining a lattice and algebraic operations $\oplus$ (alternative causes), $\star$ (joint causation) and $\cdot$ (rule application).
Main ideas

- **Multi-valued** semantics for logic programs: each true atom will be associated to a set of justifications (*causal graphs*).

- Accordingly, falsity = lack of justification.
  - This coincides with the informal reading for default negation: \( \text{not } p = \text{there is no way to derive } p \)

- Causes must be **non-redundant**.
  - Some causes will be stronger than others.
  - This allows us defining a lattice and algebraic operations \( + \) (alternative causes), \( * \) (joint causation) and \( \cdot \) (rule application).

- Important result: **semantically obtained causal values** correspond to (non-redundant) **syntactic proofs** using the program rules!
Outline

1 Motivation and examples
2 Causes as graphs
3 Positive programs
4 Default negation
5 Queries about causality
6 Conclusions and future work
**Definition**

A causal graph $G$ is a *transitively and reflexively closed* graph of labels.
Definition

A causal graph $G$ is a \textit{transitively and reflexively closed} graph of labels.

In our example, we would actually have
Definition

A causal graph $G$ is a transitive and reflexively closed graph of labels.

In our example, we would actually have

\[
\begin{array}{c}
d \\
\downarrow \\
e \\
\downarrow \\
l \\
\downarrow \\
k \\
\downarrow \\
d
\end{array}
\]
**Causal Graphs**

- $G^*$ is the transitive and reflexive closure of $G$
- **Product** $G \cdot G' \overset{\text{def}}{=} (G \cup G')^*$
- **Application** $G \cdot G' \overset{\text{def}}{=} \text{graph with vertices } V \cup V' \text{ and edges } E \cup E' \cup \{ (x, y) | x \in V, y \in V' \}$
- **Atomic graphs** $\ell$ stands for $\langle \{\ell\}, \{(\ell, \ell)\} \rangle$

Any causal graph can be built from product, application and atomic graphs. Example:

\[
d \rightarrow k \leftarrow \ell \downarrow e (d^* k) \cdot \ell \cdot e
\]
Causal Graphs

- $G^*$ is the transitive and reflexive closure of $G$

- Product $G \ast G' \overset{\text{def}}{=} (G \cup G')^*$

- Application $G \cdot G' \overset{\text{def}}{=} \text{graph with vertices } V \cup V' \text{ and edges } E \cup E' \cup \{(x, y) \mid x \in V, y \in V'\}$

- Atomic graphs $\ell$ stands for $\langle\{\ell\}, \{(\ell, \ell)\}\rangle$

- Any causal graph can be built from product, application and atomic graphs. Example:

$$
\begin{array}{c}
d \\
\downarrow \ell \\
e \\
\downarrow \\
k \\
\downarrow \\
\ell
\end{array}
$$

$$(d \ast k) \cdot \ell \cdot e$$
Causal Graphs

Definition

A causal graph $G$ is sufficient for (or weaker than) another causal graph $G'$, written $G \leq G'$, when $G \supseteq G'$.

- Notice that direction is switched: the smaller the graph, the stronger the cause!
Causal Graphs

**Definition**

A causal graph $G$ is **sufficient** for (or **weaker** than) another causal graph $G'$, written $G \leq G'$, when $G \supseteq G'$.

- Notice that **direction is switched**: the smaller the graph, the **stronger** the cause!

- The empty graph $\langle \emptyset, \emptyset \rangle$ is the top element, denoted by $\top$.
  - $\top$ stands for absolute truth, and assigned to $\top$.
  - $1$ is the $\ast$ product and $\cdot$ application identity $t \ast 1 = t$ and $t \cdot 1 = 1 \cdot t = t$. 

---

J. Fandinno  
1- University of Corunna, SPAIN  
2- Vienna University of Technology, AUSTRIA 

Causal stable models  
February 24th, 2015  
9 / 35
A causal graph $G$ is sufficient for (or weaker than) another causal graph $G'$, written $G \leq G'$, when $G \supseteq G'$. 

- Notice that direction is switched: the smaller the graph, the stronger the cause!

- The empty graph $\langle \emptyset, \emptyset \rangle$ is the top element, denoted by $1$.
  - stands for absolute truth, and assigned to $\top$.
  - $1$ is the $\ast$ product and $\cdot$ application identity $t \ast 1 = t$ and $t \cdot 1 = 1 \cdot t = t$

- We add a bottom element $0$,
  - weaker than any causal graph $0 < G$ for all $G$,
  - stands for false,
  - $0$ is the $\ast$ and $\cdot$ application annihilator $t \ast 0 = 0$ and $t \cdot 0 = 0 \cdot t = 0$
Positive programs

- Syntax: as usual plus an (optional) rule label

\[ t : H \leftarrow B_1, \ldots, B_n \]

with \( H, B_i \) atoms and \( t \) can be a label \( t = \ell \) or \( t = 1 \).
Positive programs

- Syntax: as usual plus an (optional) rule label

\[ t : H \leftarrow B_1, \ldots, B_n \]

with \( H, B_i \) atoms and \( t \) can be a label \( t = \ell \) or \( t = 1 \).

Definition (Causal model)

A causal model of \( P \) is an interpretation such that, for each rule:

\[ (\mathcal{I}(B_1) \ast \cdots \ast \mathcal{I}(B_n)) \cdot t \leq \mathcal{I}(H) \]
Alternative causes (symmetrical overdetermination)

Example

- A second law $m$ specifies that *resisting* to authority is *punishable*.
- Suppose that some person drove drunk and resisted to authority.

\[
\ell : \text{punish} \leftarrow \text{drive}, \text{drunk} \quad d : \text{drive} \quad k : \text{drunk}
\]

\[
e : \text{prison} \leftarrow \text{punish} \quad m : \text{punish} \leftarrow \text{resist} \quad r : \text{resist}
\]

- Two equally valid alternative causes

\[
(d \cdot k) \cdot \ell \cdot e \\
\]

\[
r \cdot m \cdot e
\]
Alternative causes: Addition

- addition (+) represents alternative causes

\[ \mathcal{I}(\text{punish}) = (d \times k) \cdot \ell + r \cdot m \cdot e \]

- Causal values are ideals of causal graphs. (+) corresponds to the union (\( \cup \)) of ideals.

- Disregard redundant causes.
Alternative causes

**Theorem**

\[ \langle V_{Lb}, +, *, \cdot \rangle \] is the free algebra generated by labels \( Lb \). Operations \( * \) and \(+\) are the meet and join of a completely distributive lattice.

<table>
<thead>
<tr>
<th>Property</th>
<th>Equation 1</th>
<th>Equation 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Associativity</strong></td>
<td>( t + (u+w) = (t+u) + w )</td>
<td>( t * (u<em>w) = (t</em>u) * w )</td>
</tr>
<tr>
<td><strong>Commutativity</strong></td>
<td>( t + u = u + t )</td>
<td>( t * u = u * t )</td>
</tr>
<tr>
<td><strong>Absorption</strong></td>
<td>( t = t + (t*u) )</td>
<td>( t = t * (t+u) )</td>
</tr>
<tr>
<td><strong>Distributive</strong></td>
<td>( t + (u*w) = (t+u) * (t+w) )</td>
<td>( t * (u+w) = (t<em>u) + (t</em>w) )</td>
</tr>
<tr>
<td><strong>Identity</strong></td>
<td>( t = t + 0 )</td>
<td>( t = t * 1 )</td>
</tr>
<tr>
<td><strong>Annihilator</strong></td>
<td>( 1 = 1 + t )</td>
<td>( 0 = 0 * t )</td>
</tr>
</tbody>
</table>
Alternative causes

- More specific are the ($\cdot$) application equations

<table>
<thead>
<tr>
<th>Associativity</th>
<th>Addition distributivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t \cdot (u \cdot w) = (t \cdot u) \cdot w$</td>
<td>$t \cdot (u+w) = (t \cdot u) + (t \cdot w)$</td>
</tr>
<tr>
<td>($t + u) \cdot w = (t \cdot w) + (u \cdot w)$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Identity</th>
<th>Annihilator</th>
<th>Absorption</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t = t \cdot 1$</td>
<td>$0 = t \cdot 0$</td>
<td>$t = t + u \cdot t \cdot w$</td>
</tr>
<tr>
<td>$t = 1 \cdot t$</td>
<td>$0 = 0 \cdot t$</td>
<td>$u \cdot t \cdot w = t \cdot u \cdot t \cdot w$</td>
</tr>
</tbody>
</table>

- $l$ is a label, $c$, $d$ and $e$ terms without (+)

<table>
<thead>
<tr>
<th>Label idempotence</th>
<th>Product distributivity</th>
<th>Transitivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l \cdot l = l$</td>
<td>$c \cdot (d \cdot e) = (c \cdot d) \cdot (c \cdot e)$</td>
<td>$c \cdot d \cdot e = (c \cdot d) \cdot (d \cdot e)$ with $d \neq 1$</td>
</tr>
<tr>
<td>($c \cdot d) \cdot e = (c \cdot e) \cdot (d \cdot e)$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

J. Fandinno

Causal stable models

February 24th, 2015
Positive programs

Definition (Direct consequences)

\[ T_P(\mathcal{I})(p) \overset{\text{def}}{=} \sum \left\{ \left( \mathcal{I}(B_1) \ast \ldots \ast \mathcal{I}(B_n) \right) \cdot t \mid (t : p \leftarrow B_1, \ldots, B_n) \in P \right\} \]

Theorem (Analogous to standard LP)

Let \( P \) be a (possibly infinite) positive logic program with \( n \) causal rules.

(i) \( \text{lfp}(T_P) \) is the least model of \( P \),

(ii) \( \text{lfp}(T_P) = T_P \uparrow^\omega (\emptyset) \), and

(iii) iteration ends in finite steps when \( P \) is finite \( \text{lfp}(T_P) = T_P \uparrow^n (\emptyset) \).

Theorem

Removing all labels we get the traditional (two-valued) least model.
Positive programs

- Positive programs have a least model.

\[ I(\text{prison}) = (d \ast k) \cdot \ell \cdot e + r \cdot m \cdot e \]

- If we remove all labels, then it corresponds to the standard least model.

\[ I(\text{prison}) = 1 \]
Positive programs

- Positive programs have a least model.
  \[ \mathcal{I}(\text{prison}) = (d \times k) \cdot \ell \cdot e + r \cdot m \cdot e \]

- If we remove all labels, then it corresponds to the standard least model.
  \[ \mathcal{I}(\text{prison}) = 1 \]

- Each subterm with no sums is a cause. But what do causal values really capture?
Positive programs have a least model.

\[ I(\text{prison}) = (d \ast k) \cdot \ell \cdot e + r \cdot m \cdot e \]

If we remove all labels, then it corresponds to the standard least model.

\[ I(\text{prison}) = 1 \]

Each subterm with no sums is a cause. But what do causal values really capture?

- syntactic proofs?

Notice we have not used syntactic information!
Positive programs

- Positive programs have a least model.
  \[ \mathcal{I}(\text{prison}) = (d \ast k) \cdot l \cdot e + r \cdot m \cdot e \]

- If we remove all labels, then it corresponds to the standard least model.
  \[ \mathcal{I}(\text{prison}) = 1 \]

- Each subterm with no sums is a cause. But what do causal values really capture?
  ▶ syntactic proofs?
  ▶ some proofs? all proofs?

- Notice we have not used syntactic information!
**Theorem**

*The causal value of an atom in the least model *exactly corresponds to all its possible (non-redundant) proofs.*

\[
\begin{align*}
\text{drive}(d) & \quad \text{drunk}(k) \\
\text{punish}(\ell) & \\
\text{prison}(e) & \\
\end{align*}
\]

\[
\begin{align*}
\text{resist}(r) & \quad \text{punish}(m) \\
\text{prison}(e) & \\
\end{align*}
\]
Outline

1. Motivation and examples
2. Causes as graphs
3. Positive programs
4. Default negation
5. Queries about causality
6. Conclusions and future work
Default negation

- Negation will be used for representing defaults.
  - Inertia laws are an example of dynamic defaults.
Negation will be used for representing defaults.

- Inertia laws are an example of dynamic defaults.
- Suppose now that we add time to our running example and we are imprisoned by resist at situation $s_1$, then

\[ \text{Inertia law} \]

\[ \text{prison}(T + 1) \leftarrow \text{prison}(T), \not\text{free}(T + 1) \]
Negation will be used for representing defaults.

- Inertia laws are an example of dynamic defaults.
- Suppose now that we add time to our running example and we are imprisoned by resist at situation \( s_1 \), then

\[
\text{Inertia law}
\]

\[
\text{prison}(T + 1) \leftarrow \text{prison}(T), \text{ not free}(T + 1)
\]
Negation will be used for representing defaults.
- Inertia laws are an example of dynamic defaults.
- Suppose now that we add time to our running example and we are imprisoned by resist at situation $s_1$, then

Inertia law

$$\text{prison}(T + 1) \iff \text{prison}(T), \ not \ free(T + 1)$$

Causal values persist by inertia. We disregard explanations for not being free along that period!
Default negation

- \( \text{not free}(T + 1) \) is the default (or expected) behaviour
  - if this happens, no cause is propagated (\( \text{not free}(T + 1) \) becomes 1).
Default negation

- \textit{not free}(T + 1) is the default (or expected) behaviour
  - if this happens, no cause is propagated (\textit{not free}(T + 1) becomes 1).

- Program reduct.
  - **Static default**: punished people normally goes to prison

\begin{align*}
  l : & \quad \text{punish} \leftarrow \text{drive, drunk} & d : & \quad \text{drive} \\
  m : & \quad \text{punish} \leftarrow \text{resist} & k : & \quad \text{drunk} \\
  e : & \quad \text{prison} \leftarrow \text{punish, not abnormal} & r : & \quad \text{resist}
\end{align*}
Default negation

- \textit{not free}(T + 1) is the default (or expected) behaviour
  - if this happens, no cause is propagated (\textit{not free}(T + 1) becomes 1).

- Program reduct.
  Static default: punished people normally goes to prison

\[
\begin{align*}
\ell : & \quad \text{punish} \leftarrow \text{drive, drunk} & \\
\m : & \quad \text{punish} \leftarrow \text{resist} & \\
\e : & \quad \text{prison} \leftarrow \text{punish, not abnormal} & \\
\d : & \quad \text{drive} & \\
\k : & \quad \text{drunk} & \\
\r : & \quad \text{resist} \\
\end{align*}
\]

- If we assume $I(\text{abnormal}) = 0$ (false).

\[
\begin{align*}
\ell : & \quad \text{punish} \leftarrow \text{drive, drunk} & \\
\m : & \quad \text{punish} \leftarrow \text{resist} & \\
\e : & \quad \text{prison} \leftarrow \text{punish, not abnormal} & \\
\d : & \quad \text{drive} & \\
\k : & \quad \text{drunk} & \\
\r : & \quad \text{resist} \\
\end{align*}
\]
Default negation

- we can flexibly add exceptions
  
  $\text{abnormal} \leftarrow \text{pardon}$
  $\text{abnormal} \leftarrow \text{revoke}$
  $\text{abnormal} \leftarrow \text{diplomat}$

- If we assume to be a diplomat, then $I(\text{abnormal}) = 1$ (true).

  $\ell: \text{punish} \leftarrow \text{drive}, \text{drunk}$
  $\text{m}: \text{punish} \leftarrow \text{resist}$
  $\text{e}: \text{prison} \leftarrow \text{punish, not abnormal}$
  $\text{d}: \text{drive}$
  $\text{k}: \text{drunk}$
  $\text{r}: \text{resist}$

Theorem

For each (standard) two-valued stable model there is (exactly one) corresponding causal stable model and vice versa.
1. Motivation and examples
2. Causes as graphs
3. Positive programs
4. Default negation
5. Queries about causality
6. Conclusions and future work
Sufficient Cause

Why are we in prison?

▶ *sufficient*(*X*, *prison*)?, *X* should be a minimal explanation

\[
\begin{align*}
(d \ast k) \cdot \ell \cdot e &= r \cdot m \cdot e
\end{align*}
\]
Why are we in prison?

- \textit{sufficient}(X, \textit{prison})?, $X$ should be a minimal explanation

- Was $d * k * \textit{chew}$ sufficient to cause it?
- \textit{sufficient}(d * k * \textit{chew}, \textit{prison}) should holds, despite of lack of minimality
Sufficient Cause

Given a causal graph $G$

- $G$ is a sufficient explanation for $p$ iff $G \leq I(p)$

- $G$ is a sufficient cause for $p$ iff $G$ is a subgraph-minimal sufficient explanation for $p$
Given a causal graph $G$

- $G$ is a sufficient explanation for $p$ iff $G \leq l(p)$
- $G$ is a sufficient cause for $p$ iff $G$ is a subgraph-minimal sufficient explanation for $p$

Complexity (complete results)

<table>
<thead>
<tr>
<th>positive</th>
<th>well founded</th>
<th>answer set (brave)</th>
<th>answer set (cautions)</th>
</tr>
</thead>
<tbody>
<tr>
<td>entailment</td>
<td>$P$</td>
<td>$P$</td>
<td>NP</td>
</tr>
<tr>
<td>explanation</td>
<td>$P$</td>
<td>$P$</td>
<td>NP</td>
</tr>
<tr>
<td>cause</td>
<td>$P$</td>
<td>$P$</td>
<td>NP</td>
</tr>
</tbody>
</table>

- same complexity than entailment in standard LP
Necessary Cause

Why are we in prison?
  ▶ What has been necessary to cause it?

\[(d \ast k) \cdot \ell \cdot e\]

\[r \cdot m \cdot e\]
Necessary Cause

- Why are we in prison?
  - What has been necessary to cause it?

\[
\begin{align*}
(d \ast k) \cdot \ell \cdot e &= r \cdot m \cdot e \\
\end{align*}
\]

- Only the rule \( e \) has been necessary.
Why are we in prison?

What has been necessary to cause it?

\[(d * k) \cdot \ell \cdot e\]

Only the rule \(e\) has been necessary.

Suppose we do not resit. Then \(\textit{drive}\) and \(\textit{drunk}\) would have been necessary causes.

Suppose we were not drunk. Then \(\textit{resit}\) would have been a necessary cause.
Necessary Cause

Given a causal graph $G$

- $G$ is a necessary cause for $p$ iff $G$ subgraph of all sufficient causes for $p$ and $I(p) \neq 0$
- $G$ is a necessary cause for $p$ iff $G \geq I(p)$ and $I(p) \neq 0$

Complexity (complete results)

<table>
<thead>
<tr>
<th></th>
<th>positive</th>
<th>well founded</th>
<th>answer set (brave)</th>
<th>answer set (cautions)</th>
</tr>
</thead>
<tbody>
<tr>
<td>entailment</td>
<td>P</td>
<td>P</td>
<td>NP</td>
<td>coNP</td>
</tr>
<tr>
<td>necessary</td>
<td>coNP</td>
<td>coNP</td>
<td>$\Sigma^p_2$</td>
<td>coNP</td>
</tr>
</tbody>
</table>
Why are we in prison?

- Actual Cause ≈ contingency necessary cause.

- There exists a possible world where $G$ is a necessary cause [Pearl 2000, Halpern & Pearl 2001 and 2005].

\[
\begin{align*}
(d * k) \cdot \ell \cdot e & \quad \quad \quad \quad r \cdot m \cdot e
\end{align*}
\]
Why are we in prison?

- Actual Cause ≈ contingency necessary cause.

- There exists a possible world where $G$ is a necessary cause [Pearl 2000, Halpern & Pearl 2001 and 2005].

- Contributory cause: Necessary condition in a sufficient cause [Mackie 1965, Wright 1988]
**Actual and Contributory Cause**

- **Given a causal graph** $G$
  - $G$ is a actual cause for $p$ iff there exists a sufficient cause $G'$ for $p$ such that $G \subseteq G'$

- **Complexity**

<table>
<thead>
<tr>
<th></th>
<th>positive</th>
<th>well founded</th>
<th>answer set (brave)</th>
<th>answer set (cautions)</th>
</tr>
</thead>
<tbody>
<tr>
<td>entailment</td>
<td>$P$</td>
<td>$P$</td>
<td>NP</td>
<td>coNP</td>
</tr>
<tr>
<td>actual</td>
<td>$\leq$ NP</td>
<td>$\leq$ NP</td>
<td>$\leq$ NP</td>
<td>$\leq \Pi_2^P$</td>
</tr>
<tr>
<td>HP 2001</td>
<td></td>
<td></td>
<td>NP / $\Sigma_2^P$</td>
<td></td>
</tr>
<tr>
<td>HP 2005</td>
<td></td>
<td></td>
<td>$D_2^P$</td>
<td></td>
</tr>
</tbody>
</table>

- [Eiter & Lukasiewicz 2001, Aleksandrowicz et. al. 2014]
- $\Sigma_2^P \leq D_2^P \leq \Delta_3^P \leq \Sigma_3^P$
- $\Pi_2^P \leq D_2^P \leq \Delta_3^P \leq \Pi_3^P$
Example (Lewis2000)

Suzy throws a rock at a bottle. The rock hits the bottle, shattering it. Suzy’s friend Billy throws a rock at the bottle a couple of seconds later. Who has caused the bottle to shattered?

\[
\begin{align*}
\text{hit}(\text{suzy}) & = \text{throw}(\text{suzy}) \\
\text{hit}(\text{billy}) & = \text{throw}(\text{billy}) \land \neg \text{hit}(\text{suzy}) \\
\text{shattered} & = \text{hit}(\text{suzy}) \lor \text{hit}(\text{billy})
\end{align*}
\]

Suppose that John has also thrown after Billy.

\[
\begin{align*}
  \text{hit}(\text{suzy}) & = \text{throw}(\text{suzy}) \\
  \text{hit}(\text{billy}) & = \text{throw}(\text{billy}) \land \neg \text{hit}(\text{suzy}) \\
  \text{hit}(\text{john}) & = \text{throw}(\text{john}) \land \neg \text{hit}(\text{suzy}) \land \neg \text{hit}(\text{billy}) \\
  \text{shattered} & = \text{hit}(\text{suzy}) \lor \text{hit}(\text{billy}) \lor \text{hit}(\text{john})
\end{align*}
\]
Causality and Knowledge Representation

- Suppose that John has also thrown after Billy.

\[
\begin{align*}
\text{hit}(\text{suzy}) &= \text{throw}(\text{suzy}) \\
\text{hit}(\text{billy}) &= \text{throw}(\text{billy}) \land \neg \text{hit}(\text{suzy}) \\
\text{hit}(\text{john}) &= \text{throw}(\text{john}) \land \neg \text{hit}(\text{suzy}) \land \neg \text{hit}(\text{billy}) \\
\text{shattered} &= \text{hit}(\text{suzy}) \lor \text{hit}(\text{billy}) \lor \text{hit}(\text{john})
\end{align*}
\]

- Change: John has thrown **before** Suzy.
Suppose that John has also thrown after Billy.

\[
\begin{align*}
\text{hit} (\text{suzy}) &= \text{throw} (\text{suzy}) \\
\text{hit} (\text{billy}) &= \text{throw} (\text{billy}) \land \neg \text{hit} (\text{suzy}) \\
\text{hit} (\text{john}) &= \text{throw} (\text{john}) \land \neg \text{hit} (\text{suzy}) \land \neg \text{hit} (\text{billy}) \\
\text{shattered} &= \text{hit} (\text{suzy}) \lor \text{hit} (\text{billy}) \lor \text{hit} (\text{john})
\end{align*}
\]

Change: John has thrown before Suzy.

\[
\begin{align*}
\text{hit} (\text{suzy}) &= \text{throw} (\text{suzy}) \land \neg \text{hit} (\text{john}) \\
\text{hit} (\text{billy}) &= \text{throw} (\text{billy}) \land \neg \text{hit} (\text{suzy}) \land \neg \text{hit} (\text{john}) \\
\text{hit} (\text{john}) &= \text{throw} (\text{john}) \\
\text{shattered} &= \text{hit} (\text{suzy}) \lor \text{hit} (\text{billy}) \lor \text{hit} (\text{john})
\end{align*}
\]
Causality and Knowledge Representation

- Suppose that John has also thrown after Billy.

\[
\begin{align*}
\text{hit}(\text{suzy}) &= \text{throw}(\text{suzy}) \\
\text{hit}(\text{billy}) &= \text{throw}(\text{billy}) \land \neg\text{hit}(\text{suzy}) \\
\text{hit}(\text{john}) &= \text{throw}(\text{john}) \land \neg\text{hit}(\text{suzy}) \land \neg\text{hit}(\text{billy}) \\
\text{shattered} &= \text{hit}(\text{suzy}) \lor \text{hit}(\text{billy}) \lor \text{hit}(\text{john})
\end{align*}
\]

- Change: John has thrown before Suzy.

\[
\begin{align*}
\text{hit}(\text{suzy}) &= \text{throw}(\text{suzy}) \land \neg\text{hit}(\text{john}) \\
\text{hit}(\text{billy}) &= \text{throw}(\text{billy}) \land \neg\text{hit}(\text{suzy}) \land \neg\text{hit}(\text{john}) \\
\text{hit}(\text{john}) &= \text{throw}(\text{john}) \\
\text{shattered} &= \text{hit}(\text{suzy}) \lor \text{hit}(\text{billy}) \lor \text{hit}(\text{john})
\end{align*}
\]

- Small changes implies revise the entire model. Problem of tolerance to the elaboration [McCarthy1998]
Example (Lewis2000)

Suzy throws a rock at a bottle. The rock hits the bottle, shattering it. Suzy’s friend Billy throws a rock at the bottle a couple of seconds later. Who has caused the bottle to shattered?

\[
\begin{align*}
\text{shattered}(T+1) &\leftarrow \text{throws}(X,T), \text{ not shattered}(T) \\
\text{throw}(\text{suzy},2)
\end{align*}
\]

\[
\begin{align*}
\text{throw}(\text{billy},4)
\end{align*}
\]
Example (Lewis2000)

Suzy throws a rock at a bottle. The rock hits the bottle, shattering it. Suzy’s friend Billy throws a rock at the bottle a couple of seconds later. Who has caused the bottle to shattered?

\[
\text{shattered}(T + 1) \leftarrow \text{throws}(X, T), \ not \ \text{shattered}(T)
\]

\[
\text{throw}(suzy, 2)
\]

\[
\text{throw}(billy, 4)
\]

- Inertia axiom

\[
\text{shattered}(T + 1) \leftarrow \text{shattered}(T)
\]

- We may conclude that the bottle is \textit{shattered} at 3, but not who caused it.
$r_1 : shattered(T + 1) \leftarrow throws(X, T), \text{ not shattered}(T)$

$suzy : throw(suzy, 2)$

$billy : throw(billy, 4)$
\[ r_1 : shattered(T + 1) \leftarrow throws(X, T), \text{not shattered}(T) \]

\[ suzy : \text{throw}(suzy, 2) \]

\[ billy : \text{throw}(billy, 4) \]

- We may conclude that the bottle is *shattered* at 3 because

\[
\text{suzy} \downarrow \text{suzy} \cdot r_1
\]

- Note that rule \( r_1 \) for \( T = 4 \) is not in the reduct of the program

---
Conclusions

- Multi-valued semantics based on (ideals of) causal graphs
- Values capture non-redundant proofs, but with semantic, algebraic operations
- Default negation = absence of cause.
  - Reduct definition allows defining causal stable models
  - Allows expressing dynamic defaults (ex: inertia laws)
- Ongoing work:
  - Studying actual causation.
  - Adding this causal operators on rule bodies.
Causal Graph Justifications of Stable Models

Jorge Fandinno

Thanks for your attention!

Logical Reasoning and computation
Corunna, Spain

February 24th, 2015