The Factual Counterfactual Counter

Evante Garza-Licudine, Malvin Gattinger

The idea

Remember the Hamburger example from homework II. Could we get a computer to do it for us?
Introduction
  The MCA framework
  Updates
  Retraction

Implementation
  Modelling
  Semantics
  Updates
  Retraction
  Retraction

Examples
  Hansson’s Hamburger
  Cheese and Onion

Conclusion
  Results
  Future work
Introduction
This paper provides an update semantics for counterfactual conditionals. It does so by giving a dynamic twist to the ‘Premise Semantics’ for counterfactuals developed in Veltman (1976) and Kratzer (1981).

F. Veltman: *Making Counterfactual Assumptions*
A cognitive state $S = \langle F_S, U_S \rangle$ is a list of worlds:

<table>
<thead>
<tr>
<th></th>
<th>$q$</th>
<th>$p$</th>
<th>$r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_0$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$w_1$</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$w_2$</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$w_3$</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$w_4$</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$w_5$</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$w_6$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$w_7$</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

We denote being in $F_S$ by a line $|$ left of the world. Worlds are not in $U_S$ iff they are stroked out.
We can update a cognitive state with a *fact*/observation:

\[
\begin{array}{c|ccc}
& q & p & r \\
\hline
w_0 & 0 & 0 & 0 \\
w_1 & 0 & 0 & 1 \\
w_2 & 1 & 0 & 0 \\
w_3 & 1 & 0 & 1 \\
w_4 & 0 & 1 & 0 \\
w_5 & 0 & 1 & 1 \\
w_6 & 1 & 1 & 0 \\
w_7 & 1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{c|ccc}
& q & p & r \\
\hline
w_0 & 0 & 0 & 0 \\
w_1 & 0 & 0 & 1 \\
w_2 & 1 & 0 & 0 \\
w_3 & 1 & 0 & 1 \\
w_4 & 0 & 1 & 0 \\
w_5 & 0 & 1 & 1 \\
w_6 & 1 & 1 & 0 \\
w_7 & 1 & 1 & 1 \\
\end{array}
\]

\[ [q \lor \neg r] = \]

Updating with a fact only changes $F_S$. 

We can update a cognitive state with a *fact*/observation:
We can update a cognitive state with a law:

<table>
<thead>
<tr>
<th></th>
<th>q</th>
<th>p</th>
<th>r</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_0$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$w_1$</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$w_2$</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$w_3$</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$w_4$</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$w_5$</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$w_6$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$w_7$</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$\square(p \rightarrow (q \lor r))$ =

<table>
<thead>
<tr>
<th></th>
<th>q</th>
<th>p</th>
<th>r</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_0$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$w_1$</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$w_2$</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$w_3$</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$w_4$</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$w_5$</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$w_6$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$w_7$</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Updating with a law deletes all worlds in which it is false from both $F_S$ and $U_S$. 

We can update a cognitive state with a law:
Retraction
Defining a Basis

Definition 3 (Basis) Let $S = \langle U_S, F_S \rangle$ be a state.

(i) The situation $s$ forces the proposition $P$ within $U_S$ iff for every $w \in U_S$ such that $s \subseteq w$ it holds that $w \in P$.

(ii) The situation $s$ determines the world $w$ iff $s$ forces $\{w\}$ within $U_S$.

(iii) The situation $s$ is a basis for the world $w$ iff $s$ is a minimal situation determining $w$ within $U_S$. 
Retraction
Retracting Worlds and cognitive states

**Definition 4 (Retraction)** Let $S = \langle U_S, F_S \rangle$ be a state.

(i) Suppose $w \in U_S$, and $P \subseteq W$. The set $w \downarrow P$ is determined as follows:

$s \in w \downarrow P$ iff $s \subseteq w$ and there is a basis $s'$ for $w$ such that $s$ is a maximal subset of $s'$ not forcing $P$.

(ii) $S \downarrow P$, the retraction of $P$ from $S$, is the state $\langle U_{S\downarrow P}, F_{S\downarrow P} \rangle$ determined as follows:

(a) $w \in U_{S\downarrow P}$ iff $w \in U_S$
(b) $w \in F_{S\downarrow P}$ iff $w \in U_S$ and there are $w' \in F_S$ and $s \in w' \downarrow P$ such that $s \subseteq w$.

(iii) The state $S[\text{if it had been the case that } \varphi]$ is given by $(S \downarrow [\neg \varphi])[\varphi]$
Implementation
The list of propositions is given as an argument to construct the neutral cognitive state:

```python
language=['p','q','r']
genworlds(language)
```

Logical constants:

```plaintext
phi="~(p)"
phi="(p) &(q)"
phi="(p) |(q)"
phi="(p) >(q)"
```

Bracket conventions
Modelling

Worlds

A world has two sub-structures:

```
{
    'meta': { 'FS': True, 'US': True, 'name': 'w_3' },
    'values': { 'p': 1, 'q': 0, 'r': 1 }
},
```

This one corresponds to this line in a table:

<table>
<thead>
<tr>
<th></th>
<th>p</th>
<th>q</th>
<th>r</th>
</tr>
</thead>
<tbody>
<tr>
<td>w₃</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
A cognitive state is an array of worlds:

```json
[  
  {'meta': {'FS': True, 'US': True, 'name': 'w_0'},
   'values': {'p': 0, 'q': 0}},
  {'meta': {'FS': True, 'US': True, 'name': 'w_1'},
   'values': {'p': 0, 'q': 1}},
  {'meta': {'FS': True, 'US': True, 'name': 'w_2'},
   'values': {'p': 1, 'q': 0}},
  {'meta': {'FS': True, 'US': True, 'name': 'w_3'},
   'values': {'p': 1, 'q': 1}}
]
```
Semantics

We use a recursive function to check if a formula is true in a certain world:

```python
def tiw(world, formula):
    if len(formula) == 1:  # atomic
        return world["values"][formula]
    else:
        structure = chop(formula)
        if structure["connective"] == "~":  # negation
            return not tiw(world, structure["subright"])
        if structure["connective"] == "&":  # conjunction
            return (tiw(world, structure["subleft"])) & tiw(world, structure["subright"])  )
        if structure["connective"] == "|":  # disjunction
            return (tiw(world, structure["subleft"]) | tiw(world, structure["subright"]))  )
        if structure["connective"] ==">":  # disjunction
            return (tiw(world, structure["subleft"]) <= tiw(world, structure["subright"]))  )
```

Updating with a fact

\[ S[\phi] = \langle U_S, F_S \cap \llbracket \phi \rrbracket \rangle \text{ if } F_S \cap \llbracket \phi \rrbracket \neq \emptyset; \]
\[ S[\phi] = 0, \text{ otherwise.} \]

```python
def updateFormula(cogstate, formula):
    newstate = []
    if formulaIsConsistent(cogstate, formula):
        for world in cogstate:
            if not formulaIsTrue(world, formula):
                world[meta][FS] = False
                newstate.append(world)
    else:
        newstate = destroyAllWorlds(cogstate)
    return newstate
```
Updating with a law

\[ S[\Box \phi] = \langle U_S \cap \llbracket \phi \rrbracket, F_S \cap \llbracket \phi \rrbracket \rangle \text{ if } F_S \cap \llbracket \phi \rrbracket \neq \emptyset; \]
\[ S[\Box \phi] = 0, \text{ otherwise.} \]

```python
def updateLaw(cogstate, law):
    newstate = []
    if formulaIsConsistent(cogstate, law):
        for world in cogstate:
            if not formulaIsTrue(world, law):
                world[meta][FS] = False
                world[meta][US] = False
                newstate.append(world)
    else:
        newstate = destroyAllWorlds(cogstate)
    return newstate
```
Retraction
Of a World

We have functions to check if a situation forces, determines or is a basis. Then we can compute the set $w \downarrow P$:

$$w \downarrow P = \{ s \subseteq w \mid s \not\models P \land \exists s' \text{ basis for } w : s \subseteq_{\text{max*}} s' \}$$

```python
def retractOnWorld(cogstate, worldname, proposition):
    result = []
    world = getWorldByName(worldname, cogstate)
    for situation in sitgen(world):  # s
        if Forceable(situation, proposition, cogstate):
            continue  # s may not force P
        adding = False
        for basis in getAllBases(world, cogstate):  # s'
            if not subset(situation, basis):
                continue  # s has to be a subset of s'
            Maximal = True
            for t in subsitgen(basis):
                if Forceable(situation, proposition, cogstate):
                    continue  # t may not force P
                if subset(situation, t):
                    if situation != t:
                        Maximal = False
                    if not Maximal:
                        continue  # s should be a maximal subset of s'
                    adding = True
            if adding:
                result.append(situation)
    return result
```
Retraction
Of a State

Retracting a state boils down to retracting all worlds in $F_S$:

$$U_{S \downarrow P} = U_S$$
$$F_{S \downarrow P} = \{ w \in U_S | \exists w' \in F_S : \exists s \in w' \downarrow P : s \subseteq w \}.$$
Finally, we can now assume a counterfactual:

```python
def ifItHadBeenTheCase(cogstate, formula):
    # It’s so pretty!
    return update(retract(cogstate, proposition(cogstate, lnot(formula))), formula)
```

This gives us $(S \downarrow [\neg \phi])[\phi]$. 

If it had been the case that $\phi$
Examples
Examples
Hansson’s Hamburger

```python
# Start the tex file
out = texheader("Hansson’s Hamburger puzzle", "The Factual Counterfactual Counter")

# Need propositional letters for "seeing a man walking with a hamburger", "snackbar A is open" and "snackbar B is open".
alphabet = ["p", "q", "r"]

# Now we generate the universe
W = worldgen(alphabet)
out += texify(W)

# Update with the fact that we see the man
W = updateFormula(W, "r")
out += texify(W)

# Update with the law that if we see a man with a hamburger, he must have got it at one of the snackbars
W = updateLaw(W, "((r) > ((p) | (q)))")
out += texify(W)

# Update since we see A is open
W = updateFormula(W, "p")
out += texify(W)

# Compute the counterfactual
W = ifItHadBeenTheCase(W, "~(p)")
out += texify(W)
```
The Factual Counterfactual

Evante Garza-Licudine, Malvin Gattinger

Introduction
The MCA framework
Updates
Retraction
Implementation
Modelling
Semantics
Updates
Retraction
Retraction
Examples
Hansson’s Hamburger
Cheese and Onion
Conclusion
Results
Future work

Examples

Hansson’s Hamburger

\[
\begin{array}{c|ccc}
S_0 & q & p & r \\
\hline
w_0 & 0 & 0 & 0 \\
w_1 & 0 & 0 & 1 \\
w_2 & 1 & 0 & 0 \\
w_3 & 1 & 0 & 1 \\
w_4 & 0 & 1 & 0 \\
w_5 & 0 & 1 & 1 \\
w_6 & 1 & 1 & 0 \\
w_7 & 1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{c|ccc}
S_1 & q & p & r \\
\hline
w_0 & 0 & 0 & 0 \\
w_1 & 0 & 0 & 1 \\
w_2 & 1 & 0 & 0 \\
w_3 & 1 & 0 & 1 \\
w_4 & 0 & 1 & 0 \\
w_5 & 0 & 1 & 1 \\
w_6 & 1 & 1 & 0 \\
w_7 & 1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{c|ccc}
S_2 & q & p & r \\
\hline
w_0 & 0 & 0 & 0 \\
\top & 0 & 0 & 1 \\
w_2 & 1 & 0 & 0 \\
w_3 & 1 & 0 & 1 \\
w_4 & 0 & 1 & 0 \\
w_5 & 0 & 1 & 1 \\
w_6 & 1 & 1 & 0 \\
w_7 & 1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{c|ccc}
S_3 & q & p & r \\
\hline
w_0 & 0 & 0 & 0 \\
\top & 0 & 0 & 1 \\
w_2 & 1 & 0 & 0 \\
w_3 & 1 & 0 & 1 \\
w_4 & 0 & 1 & 0 \\
w_5 & 0 & 1 & 1 \\
w_6 & 1 & 1 & 0 \\
w_7 & 1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{c|ccc}
S_4 & q & p & r \\
\hline
w_0 & 0 & 0 & 0 \\
\top & 0 & 0 & 1 \\
w_2 & 1 & 0 & 0 \\
w_3 & 1 & 0 & 1 \\
w_4 & 0 & 1 & 0 \\
w_5 & 0 & 1 & 1 \\
w_6 & 1 & 1 & 0 \\
w_7 & 1 & 1 & 1 \\
\end{array}
\]

The last state does not support \( q \), therefore \( \neg p \leadsto q \) is not accepted in \( S_3 \). The same holds for \( \neg p \leadsto r \). But \( \neg p \leadsto q \lor r \) is accepted.
Cheese and Onion

More propositions, more questions

Increasing the possible worlds increases runtime. How much?

```python
def checkRandomCounterfactual(cogstate):
    # generate a random law and update with it:
    law = "{"+choice(alphabet)+"}>({"+choice(alphabet)+"})"
    cogstate = updateLaw(cogstate, law)

    # generate a random fact and update with it:
    fact = choice(alphabet)
    cogstate = updateFormula(cogstate, fact)

    # generate a random non-trivial counterfactual and check it:
    cfantecedent = choice(alphabet)
    restralph = list(alphabet)
    restralph.remove(cfantecedent)
    cfconsequent = choice(restralph)
    cogstateNew = ifItHadBeenTheCase(cogstate, cfantecedent)
    result = supports(cogstateNew, cfconsequent)
```

Beware: The time needed to check a counterfactual varies. To get an average result, we ran this function 1000 times on the neutral state for a given number of propositions.
Conclusion
Results

Lessons learned

- Successfully implemented the semantics from [MCA].
- Any hamburger-like example can now easily be tried.
- More than four propositions are hard to cope with.
- We can now check if interpreting counterfactuals is “just as easy as” interpreting propositional logic ...
The Veltman-curve

It is not as easy as material implication.
Future Work

- Are there further philosophical consequences?
- What about other counterfactual frameworks? Can we benchmark against Kratzer, Lewis, ... ?
- Can the complexity be removed by optimization?
- What happens in the non-classical case? Currently we hard-coded:

```python
1 truthvalues=[0,1]
```

- Predicate Logic (This would be hell.)
Got questions? Ask us!

Got counterfactuals? Go to http://tinyurl.com/counterfactual and check them!