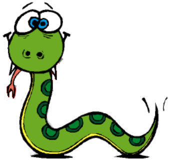
 

The Factual Counterfactual Counter

Evante Garza-Licudine, Malvin Gattinger

11.12.2012

The idea

Remember the Hamburger example from homework II.
Could we get a computer to do it for us?

Introduction

The MCA framework

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*

The MCA framework

Cognitive States

A cognitive state $S = \langle F_S, U_S \rangle$ is a list of worlds::

	q	p	r
$\vert w_0$	0	0	0
$\cancel{w_1}$	0	0	1
$\vert w_2$	1	0	0
$\vert w_3$	1	0	1
$\vert w_4$	0	1	0
w_5	0	1	1
$\vert w_6$	1	1	0
$\vert w_7$	1	1	1

We denote being in F_S by a line $|$ left of the world.
Worlds are not in U_S iff they are stroked out.

Updates
Facts

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

	q	p	r
$\mid w_0$	0	0	0
$\mid w_1$	0	0	1
$\mid w_2$	1	0	0
$\mid w_3$	1	0	1
$\mid w_4$	0	1	0
$\mid w_5$	0	1	1
$\mid w_6$	1	1	0
$\mid w_7$	1	1	1

$[q \vee \neg r] =$

	q	p	r
$\mid w_0$	0	0	0
w_1	0	0	1
$\mid w_2$	1	0	0
$\mid w_3$	1	0	1
$\mid w_4$	0	1	0
w_5	0	1	1
$\mid w_6$	1	1	0
$\mid w_7$	1	1	1

Updating with a fact only changes F_S.

Updates

Laws

We can update a cognitive state with a *law*:

	q	p	r
$\vert w_0$	0	0	0
$\vert w_1$	0	0	1
$\vert w_2$	1	0	0
$\vert w_3$	1	0	1
$\vert w_4$	0	1	0
$\vert w_5$	0	1	1
$\vert w_6$	1	1	0
$\vert w_7$	1	1	1

$$[\Box(p \to (q \vee r))] =$$

	q	p	r
$\vert w_0$	0	0	0
$\vert w_1$	0	0	1
$\vert w_2$	1	0	0
$\vert w_3$	1	0	1
$\cancel{w_4}$	0	1	0
$\vert w_5$	0	1	1
$\vert w_6$	1	1	0
$\vert w_7$	1	1	1

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

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[if it had been the case that ϕ] is given by $(S \downarrow [\![\neg\phi]\!])[\phi]$

Implementation

Modelling

Language

- The list of propositions is given as an argument to construct the neutral cognitive state:

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

- Logical constants:

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

- Bracket conventions

Modelling
Worls

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:

	p	q	r
$\vert w_3$	1	0	1

Modelling

Cognitive states

A cognitive state is an array of worlds:

```
[
  { '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:

```
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

$S[\phi] = \langle U_S, F_S \cap [\![\phi]\!] \rangle$ if $F_S \cap [\![\phi]\!] \neq \varnothing$;
$S[\phi] = \mathbf{0}$, otherwise.

```
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

$S[\Box\phi] = \langle U_S \cap [\![\phi]\!], F_S \cap [\![\phi]\!] \rangle$ if $F_S \cap [\![\phi]\!] \neq \varnothing$;
$S[\Box\phi] = \mathbf{0}$, otherwise.

```
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 $w \downarrow P$:

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

```
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 is 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:

$U_{S \downarrow P} = U_S$

$F_{S \downarrow P} = \{ w \in U_S \mid \exists w' \in F_S : \exists s \in w' \downarrow P : s \subseteq w \}.$

```
def retractOnState(cogstate,proposition):
 result=[]
 for world in cogstate:
  newworld={} # do not shoot ourselves in the foot
  newworld["values"]=dict(world["values"])
  newworld["meta"]=dict(world["meta"])
  addingToFS=False
  if world["meta"]["US"]:
   for biworld in cogstate:
    if biworld["meta"]["FS"]:
     biretract=retractOnWorld(cogstate,biworld["meta"]["name"],
          proposition)
     for s in biretract:
      if subset(s,world):
       addingToFS=True
  newworld["meta"]["FS"]=addingToFS
  result.append(dict(newworld))
 return result
```

Retraction

If it had been the case that ϕ

Finally, we can now assume a counterfactual:

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

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

Examples

Examples
Hansson's Hamburger

```
# 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)
```

Examples

Hansson's Hamburger

S_0	q	p	r
$\vert w_0$	0	0	0
$\vert w_1$	0	0	1
$\vert w_2$	1	0	0
$\vert w_3$	1	0	1
$\vert w_4$	0	1	0
$\vert w_5$	0	1	1
$\vert w_6$	1	1	0
$\vert w_7$	1	1	1

$[r] =$

S_1	q	p	r
w_0	0	0	0
$\vert w_1$	0	0	1
w_2	1	0	0
$\vert w_3$	1	0	1
w_4	0	1	0
$\vert w_5$	0	1	1
w_6	1	1	0
$\vert w_7$	1	1	1

$[\Box(r \rightarrow (p \vee q))] =$

S_2	q	p	r
w_0	0	0	0
$\cancel{w_1}$	0	0	1
w_2	1	0	0
$\vert w_3$	1	0	1
w_4	0	1	0
$\vert w_5$	0	1	1
w_6	1	1	0
$\vert w_7$	1	1	1

$S_2[p] =$

S_3	q	p	r
w_0	0	0	0
$\cancel{w_1}$	0	0	1
w_2	1	0	0
w_3	1	0	1
w_4	0	1	0
$\vert w_5$	0	1	1
w_6	1	1	0
$\vert w_7$	1	1	1

$\downarrow [\![\neg\neg p]\!][\neg p] =$

S_4	q	p	r
$\vert w_0$	0	0	0
$\cancel{w_1}$	0	0	1
w_2	1	0	0
$\vert w_3$	1	0	1
w_4	0	1	0
w_5	0	1	1
w_6	1	1	0
w_7	1	1	1

The last state does not support q, therefore $\neg p \rightsquigarrow q$ is not accepted in S_3. The same holds for $\neg p \rightsquigarrow r$.
But $\neg p \rightsquigarrow q \vee r$ *is* accepted.

Cheese and Onion

More propositions, more questions

Increasing the possible worlds increases runtime. How much?

```
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

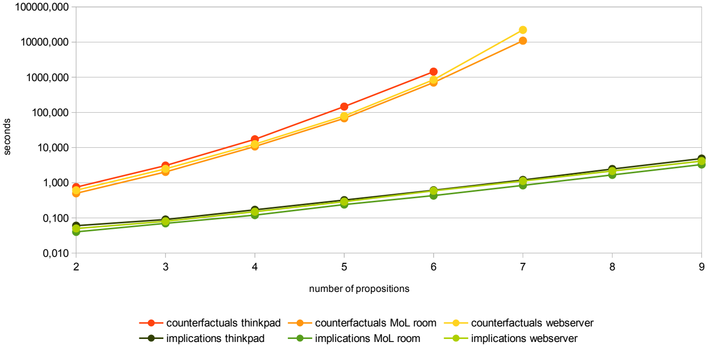
- Successfully implemented the semantics from [MCA].
- Any hamburger-like example can now easily be tried.
- More than four propositions are hard to cope with.
- We can now check if interpreting counterfactuals is "just as easy as" interpreting propositional logic ...

Results

The Veltman-curve

It is not as easy as material implication.

number of propositions:		2	3	4	5	6	7	8	9
counterfactuals	thinkpad	0,750	3,080	17,210	146,380	1442,920			
counterfactuals	MoL room	0,500	2,030	10,740	67,970	705,330	10947,400		
counterfactuals	webserver	0,610	2,480	12,500	78,760	849,320	22207,320		
implications	thinkpad	0,060	0,090	0,170	0,320	0,610	1,200	2,460	4,870
implications	MoL room	0,040	0,070	0,120	0,240	0,430	0,840	1,670	3,300
implications	webserver	0,050	0,080	0,150	0,290	0,590	1,120	2,150	4,140



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:

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

- Predicate Logic (This would be hell.)

Got questions? Ask us!

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