CL-2020 - Spiegazioni Domande

Adolfo Villafiorita

<2020-10-18 Sun>

Exercise

Translation in PL of the following sentence:

If Davide comes to the party, then, if Carlo doesn't come then Angelo comes

Mapping:

- D = Davide comes to the party
- C = Carlo comes to the party
- A = Angelo comes to the party

Candidate Solutions

Two candidate formalizations:

- F1: D imp (not C imp A)
- F2: D imp (A imp not C)

Let us check whether they are equivalent. We can compute and compare the truth tables, using:
`http://datascientia.education/logictools/prop.html`

Equivalent?

Truth Table 1

D imp (not C imp A)

```
D C A | D -> (-C -> A)
----------------------
0 0 0 |   1   1  0
0 0 1 |   1   1  1
0 1 0 |   1   0  1
0 1 1 |   1   0  1
1 0 0 |   0   1  0
1 0 1 |   1   1  1
1 1 0 |   1   0  1
1 1 1 |   1   0  1
```

Truth Table 2

D imp (A imp not C)

```
D A C | D -> (A -> -C)
----------------------
0 0 0 |   1     1  1
0 0 1 |   1     1  0
0 1 0 |   1     1  1
0 1 1 |   1     0  0
1 0 0 |   1     1  1
1 0 1 |   1     1  0
1 1 0 |   1     1  1
1 1 1 |   0     0  0
```

Equivalent? No

F1 and F2 differ in the following cases:

Row	D	A	C	F1	F2
A.	1	0	0	0	1
B.	1	1	1	1	0

Let see what they mean:
A. Davide comes, Carlo does not, and Angelo does not. This interpretation satisfies F2 (but not F1), but it is, however, "wrong", because "if Carlo does not come, then Angelo comes"; thus F2 does not formalizes our world.
B. Davide comes to the party, Carlo comes to the party and so does Angelo. This interpretation satisfies F1 (but not F2) and it is also "correct" w.r.t. the world, since if Carlo comes, nothing is said about what Angelo does.

Preliminary Remark

Peculiar case of the $\otimes$ expansion:

$$P_1 \otimes P_2 \text{corresponds to:} P_1 \vee P_2 \tag{1}$$

Exercise: Hilbert's axiom

We use the notation supported by the Logic Tools on our website:

```
(p -> (q -> r)) -> ((p -> q) -> (p -> r))
```

Assume it is false:

```
- ( (p -> (q -> r)) -> ((p -> q) -> (p -> r)) )
```

Compute the CNF

```
CNF( -((p->(q->r))->((p->q)->(p->r))) )
CNF( p->(q->r) ) & CNF( -((p->q)->(p->r)) )
[ CNF(-p) ⊗ CNF(q->r) ] & [ CNF( (p->q) ) & CNF( -(p->r) )
```

Since we have an &, we can split the decomposition.

First set of clauses

```
CNF(-p) ⊗ CNF(q->r) =
-p ⊗ [ CNF(-q) ⊗ CNF(r) ]
-p ⊗ [ -q ⊗ r ]
-p ⊗ [ -q | r ]
-p | [ -q | r ]
-p | -q | r
```

Second set of clauses

```
CNF( (p->q) )
CNF( -p ) ⊗ CNF( q ) [next time, we'll omit this step!]
-p | q
```

Third set of clauses

```
CNF( -(p->r) )
CNF(p) & CNF(-r)
p & -r
```

Back together

We can now put all the conjunctions back together:

```
(-p | -q | r) & (-p | q) & p & -r
```

Did we do it right?

We check whether the the formula and (our attempt at building) its CNF are satisfied by the same interpretations:

```
p q r | -((p -> (q -> r)) -> ((p -> q) -> (p -> r)))
--------------------------------------------------------
0 0 0 | 0     1     1     1     1     1     1
0 0 1 | 0     1     1     1     1     1     1
0 1 0 | 0     1     0     1     1     1     1
0 1 1 | 0     1     1     1     1     1     1
1 0 0 | 0     1     1     1     0     1     0
1 0 1 | 0     1     1     1     0     1     1
1 1 0 | 0     0     0     1     1     0     0
1 1 1 | 0     1     1     1     1     1     1
```

We did it right indeed!

```
p q r | ((((-p V -q) V r) & (-p V q)) & p) & -r
-----------------------------------------------
0 0 0 |     1  1 1   1    1  1  1     0    0 1
0 0 1 |     1  1 1   1    1  1  1     0    0 0
0 1 0 |     1  1 0   1    1  1  1     0    0 1
0 1 1 |     1  1 0   1    1  1  1     0    0 0
1 0 0 |     0  1 1   1    0  0  0     0    0 1
1 0 1 |     0  1 1   1    0  0  0     0    0 0
1 1 0 |     0  0 0   0    0  0  1     0    0 1
1 1 1 |     0  0 0   1    1  0  1     1    0 0
```