Experiments in SIXTEEN

M. Ultseq

July 2, 2024

The logic contains the connectives

$$\wedge_f, \wedge_t, \neg_f, \neg_t, \vee_f, \vee_t$$

and truth values

$$\mathbf{N}, \boldsymbol{N}, \boldsymbol{F}, \boldsymbol{T}, \boldsymbol{B}, \mathbf{NF}, \mathbf{NT}, \mathbf{FT}, \mathbf{NB}, \mathbf{FB}, \mathbf{TB}, \mathbf{NFT}, \mathbf{NFB}, \mathbf{NTB}, \mathbf{FTB}, \mathbf{A}.$$

The truth values $\mathbf{NT}, \mathbf{NTB}, \boldsymbol{T}, \mathbf{TB}$ are designated.

1 Finding an implication in SIXTEEN

In classical logic, implication can be defined as $\neg A \vee B$. But SIXTEEN has two versions of $\neg$ and two versions of $\vee$. For all resulting combinations, we check if the equivalents of modus ponens and the deduction theorem are valid.

Proposition 1 *The following consequence* ***does not*** *hold:*

$$A, (\neg_t A \vee_t B) \vdash B$$

Proposition 2 *The following meta-consequence* ***does not*** *hold:*

$$P, Q \vdash R \quad / \quad P \vdash (\neg_t Q \vee_t R)$$

Proposition 3 *The following consequence holds:*

$$A, (\neg_f A \vee_f B) \vdash B$$

Proposition 4 *The following meta-consequence* ***does not*** *hold:*

$$P, Q \vdash R \quad / \quad P \vdash (\neg_f Q \vee_f R)$$

Proposition 5 *The following consequence* ***does not*** *hold:*

$$A, (\neg_f A \vee_t B) \vdash B$$

Proposition 6 *The following meta-consequence* ***does not*** *hold:*

$$P, Q \vdash R \quad / \quad P \vdash (\neg_f Q \vee_t R)$$

Proposition 7 *The following consequence holds:*

$$A, (\neg_t A \vee_f B) \vdash B$$

Proposition 8 *The following meta-consequence* ***does not*** *hold:*

$$P, Q \vdash R \quad / \quad P \vdash (\neg_t Q \vee_f R)$$

2 Checking De Morgan Triples

In classical logic, $\neg$, $\vee$, and $\wedge$ satisfy De Morgan's laws. But SIXTEEN has two versions of each. For all resulting combinations, we check if the corresponding De Morgan laws are valid.

Proposition 9 *The equality* $\neg_t(A \vee_t B) = (\neg_t A \wedge_t \neg_t B)$ *holds.*

Proposition 10 *The equality* $\neg_t(A \wedge_t B) = (\neg_t A \vee_t \neg_t B)$ *holds.*

Proposition 11 *The equality* $\neg_f(A \vee_t B) = (\neg_f A \wedge_t \neg_f B)$ *does* ***not*** *hold.*

Proposition 12 *The equality* $\neg_f(A \wedge_t B) = (\neg_f A \vee_t \neg_f B)$ *does* ***not*** *hold.*

Proposition 13 *The equality* $\neg_t(A \vee_t B) = (\neg_t A \wedge_f \neg_t B)$ *does* ***not*** *hold.*

Proposition 14 *The equality* $\neg_t(A \wedge_f B) = (\neg_t A \vee_t \neg_t B)$ *does* ***not*** *hold.*

Proposition 15 *The equality* $\neg_f(A \vee_t B) = (\neg_f A \wedge_f \neg_f B)$ *does* ***not*** *hold.*

Proposition 16 *The equality* $\neg_f(A \wedge_f B) = (\neg_f A \vee_t \neg_f B)$ *does* ***not*** *hold.*

Proposition 17 *The equality* $\neg_t(A \vee_f B) = (\neg_t A \wedge_t \neg_t B)$ *does* ***not*** *hold.*

Proposition 18 *The equality* $\neg_t(A \wedge_t B) = (\neg_t A \vee_f \neg_t B)$ *does* ***not*** *hold.*

Proposition 19 *The equality* $\neg_f(A \vee_f B) = (\neg_f A \wedge_t \neg_f B)$ *does* ***not*** *hold.*

Proposition 20 *The equality* $\neg_f(A \wedge_t B) = (\neg_f A \vee_f \neg_f B)$ *does* ***not*** *hold.*

Proposition 21 *The equality* $\neg_t(A \vee_f B) = (\neg_t A \wedge_f \neg_t B)$ *does* ***not*** *hold.*

Proposition 22 *The equality* $\neg_t(A \wedge_f B) = (\neg_t A \vee_f \neg_t B)$ *does* ***not*** *hold.*

Proposition 23 *The equality* $\neg_f(A \vee_f B) = (\neg_f A \wedge_f \neg_f B)$ *holds.*

Proposition 24 *The equality* $\neg_f(A \wedge_f B) = (\neg_f A \vee_f \neg_f B)$ *holds.*

3 Program listing: `ex_sixteen.pl`

```
% Test file to check things in SIXTEEN

% make sure MUltseq is loaded
:- ensure_loaded('../multseq/multseq').

% load sample properties
:- [properties].

% load the rules
:- load_logic('shramko-wansing.msq').

% define standard Omap
:- setOmap([(negt)/(-),andt/(/\),ort/(\/)]).

% define generators

imp_ops(ort(negt(X),Y)/[X,Y]).
imp_ops(orf(negf(X),Y)/[X,Y]).
imp_ops(ort(negf(X),Y)/[X,Y]).
```

```
imp_ops(orf(negt(X),Y)/[X,Y]).

or_ops(ort(X,Y)/[X,Y]).
or_ops(orf(X,Y)/[X,Y]).

and_ops(andt(X,Y)/[X,Y]).
and_ops(andf(X,Y)/[X,Y]).

neg_ops(negt(X)/[X]).
neg_ops(negf(X)/[X]).

% auxiliary predicate callall

callall([]) :- !.
callall([P|Ps]) :-
  (call(P) ->
    callall(Ps)
  ; callall(Ps)).

% check all properties and write report to out.tex

:- set_option(tex_output(terse)).

:- start_logging(ex_sixteen,'.tex').

:- print_tex(tex_title("Experiments␣in␣SIXTEEN")).

:- print_tex(tex_logic).

:- print_tex(tex_section(["Finding␣an␣implication␣in␣SIXTEEN"])).

% check if any combination of -a \/ b satisfies modus ponens and
% deduction theorem

:- print_tex(tex_paragraph(["In␣classical␣logic,␣implication␣can␣be␣defined␣as␣$\\lnot␣A␣\\
    lor␣B$.␣But␣SIXTEEN␣has␣two␣versions␣of␣$\\lnot$␣and␣two␣versions␣of␣$\\lor$.␣For␣all␣
    resulting␣combinations,␣we␣check␣if␣the␣equivalents␣of␣modus␣ponens␣and␣the␣deduction␣
    theorem␣are␣valid."])).

:-  (property(modusponens, _, P),
      property(deductionthm, _, Q),
      instantiate(> : [P,Q] @ imp_ops, PP),
      callall(PP),
      fail)
    ; true.

:- print_tex(tex_section(["Checking␣De␣Morgan␣Triples"])).

% check all combinations of De Morgan's laws (as quasi-equations)

:- print_tex(tex_paragraph(["In␣classical␣logic,␣$\\lnot$,␣$\\lor$,␣and␣$\\land$␣satisfy␣De␣
    Morgan's␣laws.␣But␣SIXTEEN␣has␣two␣versions␣of␣each.␣For␣all␣resulting␣combinations,␣we␣
    check␣if␣the␣corresponding␣De␣Morgan␣laws␣are␣valid."])).


:-  ( property(demorganor, _, P),
      property(demorganand, _, Q),
      instantiate([\/, /\, -] : [P,Q] @ [or_ops, and_ops, neg_ops], PP),
      callall(PP),
      fail)
    ; true.

:- print_tex(tex_listing("ex_sixteen.pl")).

:- stop_logging.
```