Existential Contradiction Logic

by Sven Nilsen, 2024

In this paper I show that contradictions have different propositions in classical, existential and constructive logic. Most importantly, these propositions are not closed under existential logic, which implies that there is some existential logic for contradictions stronger than existential logic but weaker than classical logic. I call this logic for Existential Contradiction Logic.

The following table shows the weakest logic among classical, existential and constructive logic that proves theorems of the form `X => Y`, where `X, Y` are in the set `!(a & b), !(!!a & !!b), !a | !b`.

Column => Row	**!(a & b)**	**!(!!a & !!b)**	**!a | !b**
!(a & b)	IPL, refl	IPL, lemma 2	IPL, lemma 1 + 2
!(!!a & !!b)	PL, lemma 3	IPL, refl	IPL, lemma 1
!a | !b	PL, lemma 3 + 4	EL, lemma 4	IPL, refl

For lemmas, see Appendix A.

- IPL = Intuitionistic Propositional Logic (constructive)
- EL = Existential Logic (excluded middle of non-existence)
- PL = classical logic (excluded middle)

Notice that `!(a & b) => !(!!a & !!b)` is not provable in EL, but requires PL. This means that EL is not strong enough to close all contradictions. Obviously, PL closes all contradictions. However, it is desirable to have some logic which is slightly weaker than PL while also closes all contradictions.

I introduce a new axiom to close contradictions in EL:

```
!(a & b) => !(!!a & !!b)        for all `a, b` in EL
```

This logic is called Existential Contradiction Logic (ECL) and produces the following table:

Column => Row	**!(a & b)**	**!(!!a & !!b)**	**!a | !b**
!(a & b)	IPL	IPL	IPL
!(!!a & !!b)	ECL	IPL	IPL
!a | !b	ECL	EL	IPL

Appendix A

```
fn lemma_1 : !a | !b -> !(!!a & !!b) {
    x : !a | !b;

    lam r : !(!!a & !!b) {
        y1 : !!a;
        y2 : !!b;

        lam f : !a => false {
            z : !a;
            let r = y1(z) : false;
            return r;
        }
        lam g : !b => false {
            z : !b;
            let r = y2(z) : false;
            return r;
        }
        let r = match x (f, g) : false;
        return r;
    }
    return r;
}

fn lemma_2 : !(!!a & !!b) -> !(a & b) {
    use std::modus_tollens;

    x : !(!!a & !!b);

    fn f : a & b -> !!a & !!b {
        use std::not_double;
        use std::refl;

        x : a;
        y : b;

        let x2 = not_double(x) : !!a;
        let y2 = not_double(y) : !!b;
        let r = refl(x2, y2) : !!a & !!b;
        return r;
    }

    let g = modus_tollens(f) : !(!!a & !!b) => !(a & b);
    let r = g(x) : !(a & b);
    return r;
}
```

```
use std::E;
use std::Excm;

fn lemma_3 : Excm' & !(a & b) -> !(!!a & !!b) {
    use std::excm_to;

    excm : Excm';
    x : !(a & b);

    let excm_a = excm_to(excm) : a | !a;
    let excm_b = excm_to(excm) : b | !b;
    lam f : a => !(!!a & !!b) {
        y : a;

        lam f : b => !(!!a & !!b) {
            z : b;
            let z2 = x(y, z) : false;
            let r = match z2 : !(!!a & !!b);
            return r;
        }
        lam g : !b => !(!!a & !!b) {
            z : !b;

            lam r : !(!!a & !!b) {
                z2 : !!a;
                z3 : !!b;

                let r = z3(z) : false;
                return r;
            }
            return r;
        }
        let r = match excm_b (f, g) : !(!!a & !!b);
        return r;
    }
    lam g : !a => !(!!a & !!b) {
        y : !a;

        lam r : !(!!a & !!b) {
            z2 : !!a;
            z3 : !!b;

            let r = z2(y) : false;
            return r;
        }
        return r;
    }
    let r = match excm_a (f, g) : !(!!a & !!b);
    return r;
}
```

```
fn lemma_4 : E' & !(!!a & !!b) -> !a | !b {
    use std::e_to;
    use std::left;
    use std::right;

    e : E';
    x : !(!!a & !!b);

    let e_a = e_to(e) : !a | !!a;
    let e_b = e_to(e) : !b | !!b;
    let f = left() : !a => !a | !b;
    lam g : !!a => !a | !b {
        y : !!a;

        let f = right() : !b => !a | !b;
        lam g : !!b => !a | !b {
            z : !!b;

            let z2 = x(y, z) : false;
            let r = match z2 : !a | !b;
            return r;
        }
        let r = match e_b (f, g) : !a | !b;
        return r;
    }
    let r = match e_a (f, g) : !a | !b;
    return r;
}
```