Existential Logic and Graphs

by Sven Nilsen, 2024

In this paper I show that classical logic is too strong for graphs and that Existential Logic is a natural logical language for graphs as the strongest logic after relaxing classical logic.

A graph consists of nodes and edges. In this paper, I will talk about directed graphs that have maximum one edge from any node to any other node. Such graphs can be represented using square matrices of boolean values.

I want to find a natural logical language for graphs. Using the relaxing technique from Avatar Extensions, I will start with classical logic as a candidate and derive a natural logic:

    (a | !a)^true   for all `a`

Classical logic has the property of excluded middle, which means for any proposition `a`, one can construct either `a` or its negative `!a`.

In classical logic, implication corresponds to the following:

    a => b   <=>   !a | b

This means that when interpreting a graph matrix, one can attribute a single cell as implication, but one can also interpret filled rows and columns as `true` or `false` respectively, depending on whether the matrix interpretation is row or column major. Notice that along the diagonal, one gets ambiguity as `!a | a` is always true in classical logic. As a result, it is not possible to reconstruct truth values from a graph matrix without using closed world assumptions. A filled row or column can only be determined to be `true` or `false` if the matrix contains all proof relevant propositions.

The problem is that classical logic is too strong for graphs. One can see this from the theorem:

    ((a => b) | (b => a))^true   for all `a, b`

In graphs, it is natural for two nodes to not have any edge from one to the other.

Classical logic prevents the case when there is no edge between two nodes. Since graphs are natural for human brains to think about, e.g. when reasoning about causality, this undermines classical logic as a natural logic for many applications. It might seem counter-intuitive or easy to believe that it is just people thinking wrong, since classical logic is otherwise very useful. However, there is no reason to over-think this problem: Classical logic is simply not natural in this sense.

To fix this problem, one can relax excluded middle to excluded middle of negation:

    (!a | !!a)^true   for all `a`

This is Existential Logic, which is stronger than constructive logic, but weaker than classical logic.

When using Existential Logic to interpret graphs, I will continue using the same notion of implication as in classical logic `!a | b`. However, `!a | b` is not provable from `a => b` in Existential Logic. I have to focus on this notion explicitly, to keep the interpretation of rows and columns.

To make it easier to talk about `!a | b`, I will refer to it as *the relation*. No name is needed, since it is implicitly known which relation one talks about in this context, as there is only one relation.

One consequence of using Existential Logic, is that one can no longer fill the diagonal by default, since it is not possible to prove `!a | a` for all `a`. However, the relation is transitive, so there is a constraint on the graphs that can be represented. When imposing symmetry on the graph matrix, one gets a partial equivalence relation, which is important in foundational Path Semantics since path semantical quality `a ~~ b` is a partial equivalence.

A partial equivalence forms a local total equivalence by symmetry and transitivity:

(!a | b) | (!b | a) => !a | a

Notice that this only holds for graph matrices that are symmetric:

(!a | b) => (!b | a)

One can prove transitivity holds in constructive logic (here I use Hooo):

```
fn relation_transitivity : (!a | b) & (!b | c)  ->  (!a | c) {
    x : !a | b;
    y : !b | c;

    use std::left;
    use std::right;

    let x2 = left() : !a => (!a | c);
    lam x3 : b => (!a | c) {
        z : b;

        lam z2 : !b => (!a | c) {
            w : !b;

            let w2 = w(z) : false;
            let r = match w2 : !a | c;
            return r;
        }
        let z3 = right() : c => (!a | c);
        let r = match y (z2, z3) : !a | c;
        return r;
    }
    let r = match x (x2, x3) : !a | c;
    return r;
}
```

This means the natural logic is transitive, indeterminate (because the diagonal is not filled and one can not restore truth values without closed world assumptions) and not necessarily symmetric. When assuming symmetry one gets a partial equivalence relation. Existential Logic fits this language, by relaxing classical logic using a 1-avatar (negation of excluded middle).

Therefore, the state of unknown truths in a strong logic of graphs is given by Existential Logic:

(!a | !!a)^true for all `a`

This means that when reasoning about causality, one requires the analogue of Existential Logic (strong) or constructive logic (weak). The difference between these logical languages is the “tension” in the language and classical logic might be interpreted as the tension getting too strong.

Naturally, there are many consequences from this result for Analytic Philosophy (future work).

Q.E.D.