Relations and Logic: using relations*

Ian Barland
John Greiner
Phokion Kolaitis
Moshe Vardi
Matthias Felleisen

This work is produced by OpenStax-CNX and licensed under the
Creative Commons Attribution License 1.0†

Abstract

We motivate the use of relations, as a way to encapsulate the information previously spread across a swath of propositional variables.

1 Relations: Building a better (representation of) WaterWorld

So far, we have represented WaterWorld boards using propositions like A − has − 2 and B − unsafe. You've probably already felt that this is unwieldy, having hundreds propositional variables running around, with only our naming convention implying any relation between them. Worse, this zoo of propositions doesn't reflect how we actually think about WaterWorld. For instance, the only way the rules recognize that locations A and B are near each other is because of several axioms which simultaneously involve A − has − 2 and B − unsafe, *etc.*, in just the right way to result in our idea of the concept "neighbor". In fact, there is no way of talking about the location A directly; we only had propositions which dealt with its properties, such as whether or not it neighbored exactly two pirates.

If writing a program about WaterWorld, our program should reflect our conception of the problem. However, as it stands, our conception corresponds to having many many Boolean variables named A − has − 2, B − unsafe, *etc.* Even worse, the rules would be encodings of the hundreds of axioms. A long enumeration of the axioms is probably **not** how you think of the rules. In other words, when explaining the game to your friend, you probably say "if a location contains a 2, then two of its neighbors are pirates", rather than droning on for half an hour about how "if location A contains a 2, then either location B is unsafe or ...".

Moreover, the original rules only pertained to a fixed-size board; inventing a new game played on a 50×50 grid would require a whole new set of rules! That is clearly **not** how we humans conceptualize the game! What we want, when discussing the rules, is a generic way to discussing neighboring locations, so that we can have one single rule, saying that if a (generic) location has a zero, then any neighboring location is

*Version 2.25: Jan 9, 2009 10:45 am -0600
†http://creativecommons.org/licenses/by/1.0

safe. Thus, we allow the exact details of "neighboring location" to change from game to game as we play on different boards (just as which locations contain pirates changes from game to game).

In a program, you'd probably represent the board as a collection (matrix, list, whatever) of Booleans. In our logic, to correspond to this data structure, we'll introduce **binary relations**.

> ASIDE: By including relations (rather than sticking entirely with propositions), we are leaving the realm of propositional logic; we'll soon reach **first-order logic** once we also introduce quantifiers — corresponding to aspects of program control-flow (loops).

We'll start by adding a way to express whether any two locations are adjacent: a relation nhbr, which will encode the board's geography as follows: nhbr (A, B) and nhbr (Z, Y) are true, while nhbr (A, D) and nhbr (M, Z) are false.

What, exactly, do we mean by "relation"? We'll see momentarily, that we can represent nhbr as a set of pairs-of-locations (or equivalently, a function which takes in two locations, and returns either true or false.)

This relation "nhbr" entirely encodes the board's geography. Giving somebody the relation is every bit as good as to showing them a picture of the board (in some ways, better — the relation makes it perfectly clear whether two locations which just barely touch at a single point, like B and G, are meant to be considered neighbors.)

Exercise 1 ***(Solution on p. 3.)***

We used a binary (two-input) relation to describe neighboring locations. How can we use a relation to capture the notion "location A is safe"?

After defining relations and discussing their properties, we'll talk about interpreting logic formulas relative to particular relations.

Using relations gives us additional flexibility in modeling our domain, so that our formal logical model more closely corresponds to our intuition. Relations help separate the WaterWorld domain axioms (code) from the data, *i.e.*, the particular board we're playing on.

Solutions to Exercises in this Module

Solution to Exercise (p. 2)

We'll use a **unary** (one-input) relation: safe(A) is true if and only if ("iff") location A is safe.