

WATERWORLD-AXIOMS

Version 2.22: 2004/05/14 14:25:57.090 GMT-5

John Greiner
Ian Barland

This work is produced by The Connexions Project and licensed under the
Creative Commons Attribution License *

Abstract

(Blank Abstract)

waterworld-axioms

BROWSER NOTE: This page meant to be viewed with a MathML-enabled browser. If you see $(\forall x. (P(x) \rightarrow (\exists y. (P(y) \vee \phi))))$ as a nice version of (forall x . (P(x) \rightarrow (exists y . (P(y) \vee phi)))) you're doing okay; If you further see $\mathcal{A} \vdash \mathcal{B}$ as a nice version of (scriptA \vdash scriptB) you're set! If not, see our description of browser support¹.

We summarize the details of how we choose to model WaterWorld boards in propositional logic: exactly what propositions we make up, and the formal domain axioms which capture the game's rules.

The board is fixed at 6x4, named A,...,Z (with I and O omitted).

1 Propositions

There are a myriad of propositions for WaterWorld, which can be grouped:

- Whether or not a location contains a pirate: *A-unsafe*, *B-unsafe*, ..., *Z-unsafe*.
- Whether or not a location contains no pirate: *A-safe*, *B-safe*, ..., *Z-safe*.

ASIDE: Yes, using the intended interpretation these are redundant with the previous ones; some domain axioms below will formalize this.

- Propositions indicating the number of neighboring pirates, to a location: *A-has-0*, *A-has-1*, *B-has-0*, *B-has-1*, *B-has-2*, ..., *H-has-0*, *H-has-1*, *H-has-2*, *H-has-3*, ..., *Z-has-0*, *Z-has-1*. These are all true/false propositions – there are no explicit numbers in the logic. A domain axiom below will assert that whenever (say) *B-has-1* is true, then *B-has-0* and *B-has-2* are both false.

*<http://creativecommons.org/licenses/by/1.0>

¹<http://cnx.rice.edu/content/m10845/latest/>

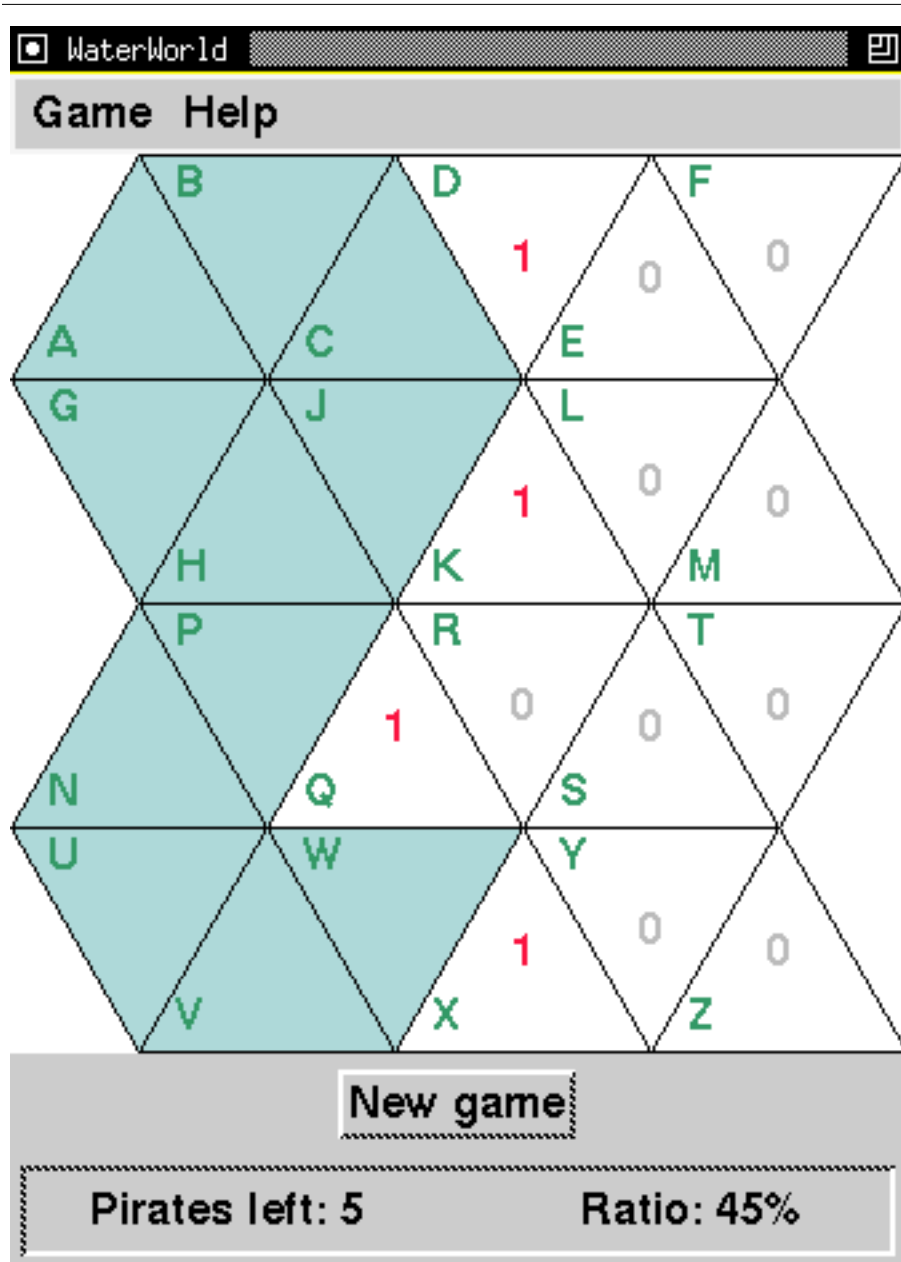


Figure 1: A Sample WaterWorld board

ASIDE: There is no proposition *A-has-2* or *A-has-3* – since location A has only one neighbor. Similarly, there is no proposition *B-has-3*. We *could* have chosen to include those, but under the intended interpretation they'd always be false.

Note that these propositions always have true/false values, regardless of whether they are ever revealed to a player: For instance, if you are playing, and location (say) A has a pirate, then you'll never be told whether or not *A-has-1* is true for that board, but it *does* still have a value.

- A propositions about the number of pirates overall on the board. *totCount-is-0*, *totCount-is-1*, ..., *totCount-is-24*. This counts both revealed and unrevealed locations, so it's not quite the same as the count displayed at the bottom of the gameboard view. In the default game, *totCount-is-5* is true, and the rest of these propositions are false.

ASIDE: Why did we choose to have propositions involving the total count, rather than the count of pirates remaining? Remember to separate a board and a *view* of the board. We could model either of these, but the underlying board is more fundamental, and that's what we choose to capture.

2 The domain axioms

- Asserting that the neighbor-count is correct:
 - Count of 0:
 - * "A0": ($A\text{-has-}0 \rightarrow (B\text{-safe} \wedge G\text{-safe}))$)
 - * ...
 - * "H0": ($H\text{-has-}0 \rightarrow (G\text{-safe} \wedge J\text{-safe} \wedge P\text{-safe}))$)
 - * ...
 - * "Z0": ($Z\text{-has-}0 \rightarrow (Y\text{-safe}))$)
 - Count of 1:
 - * "A1": ($A\text{-has-}1 \rightarrow ((B\text{-safe} \wedge G\text{-unsafe}) \vee (B\text{-unsafe} \wedge G\text{-safe}))$)
 - * ...
 - * "H1": ($H\text{-has-}1 \rightarrow ((G\text{-safe} \wedge J\text{-safe} \wedge P\text{-unsafe}) \vee (G\text{-safe} \wedge J\text{-unsafe} \wedge P\text{-safe}) \vee (G\text{-unsafe} \wedge J\text{-safe} \wedge P\text{-safe}))$)
 - * ...
 - * "Z1": ($Z\text{-has-}1 \rightarrow (Y\text{-unsafe}))$)
 - Count of 2:
 - * "A2": ($A\text{-has-}2 \rightarrow (B\text{-unsafe} \wedge F\text{-unsafe}))$)
 - * ...
 - * "H2": ($H\text{-has-}2 \rightarrow ((G\text{-safe} \wedge J\text{-unsafe} \wedge P\text{-unsafe}) \vee (G\text{-unsafe} \wedge J\text{-safe} \wedge P\text{-unsafe}) \vee (G\text{-unsafe} \wedge J\text{-unsafe} \wedge P\text{-safe}))$)
 - * ...

There aren't any such axioms for locations with only one neighbor.
 - Count of 3:
 - * "H3": ($H\text{-has-}3 \rightarrow (G\text{-unsafe} \wedge J\text{-unsafe} \wedge P\text{-unsafe}))$)
 - * ...

There aren't any such axioms for locations with only one or two neighbors.

Whew!

- Constraints relating safe and unsafe:
 - ($A\text{-safe} \rightarrow \neg A\text{-unsafe}$),
 - ($\neg A\text{-safe} \rightarrow A\text{-unsafe}$),
 - ...
 - ($Z\text{-safe} \rightarrow \neg Z\text{-unsafe}$),
 - ($\neg Z\text{-safe} \rightarrow Z\text{-unsafe}$).
- Constraints on the number of unsafe neighbors:
 - * "A" ("A has some number-of-nhbrs") ($A\text{-has-0} \vee A\text{-has-1}$)
 - * "A!" ("A has exactly 1 number-of-nhbrs") ($A\text{-has-0} \rightarrow (\neg A\text{-has-1})$)
 - * ($B\text{-has-0} \vee B\text{-has-1} \vee B\text{-has-2}$)
 - * ($B\text{-has-0} \rightarrow (\neg B\text{-has-1} \wedge \neg B\text{-has-2})$)
 - * ($B\text{-has-1} \rightarrow (\neg B\text{-has-0} \wedge \neg B\text{-has-2})$)
 - * ($B\text{-has-2} \rightarrow (\neg B\text{-has-0} \wedge \neg B\text{-has-1})$)
 - * ($H\text{-has-0} \vee H\text{-has-1} \vee H\text{-has-2} \vee H\text{-has-3}$)
 - * ($H\text{-has-0} \rightarrow (\neg H\text{-has-1} \wedge \neg H\text{-has-2} \wedge \neg H\text{-has-3})$)
 - * ($H\text{-has-1} \rightarrow (\neg H\text{-has-0} \wedge \neg H\text{-has-2} \wedge \neg H\text{-has-3})$)
 - * ($H\text{-has-2} \rightarrow (\neg H\text{-has-0} \wedge \neg H\text{-has-1} \wedge \neg H\text{-has-3})$)
 - * ($H\text{-has-3} \rightarrow (\neg H\text{-has-0} \wedge \neg H\text{-has-1} \wedge \neg H\text{-has-2})$)
- Finally, constraints about the total number of pirates:
 - If there are 5 unsafe locations total, and they are all accounted for, then every other place is safe.
 - TODO: spell these out:
 - * ..
 - * ..
 - If there are 5 unsafe locations total, 3 are accounted for, and only two are undetermined, then those two are unsafe.
 - TODO: spell these out:
 - * ..
 - * ..

As mentioned, it is a bit redundant to have propositions *A-safe* and *A-unsafe*; it just allows for an inconsistent state. If implementing this in a program, you might have both variables, but a sanity-check function to make sure that a given board representation is consistent. (Even better, these propositions wouldn't be variables, but instead they'd be calls to a lookup (accessor) function; the functions for those two propositions would examine the same internal state, to eliminate the chance of inconsistent data).

Note that using only true/false propositions (without recourse to numbers) makes this unwieldy. We'll see later how relations² and quantifiers³ help us model the game of Water-World more concisely.

²<http://cnx.rice.edu/content/m10724/latest/>

³<http://cnx.rice.edu/content/m10728/latest/>