Circuits Documentation

Release 0.0.1

Cheyenne Laue, George Lesica, Alden Wright

August 19, 2014

Contents

Contents:

CHAPTER 1

Module Documentation

1.1 technology

Technology abstraction classes and functions.

class circuits.technology.**Technology**(*name*, *circuit*, *cost=0*)

A technology, represented by a boolean circuit. Can be compared to other technologies for equality and, given a goal, closeness of approximation of the goal.

Parameters

- **name** – the name of the circuit
- **circuit** – a function array
- **cost** – the cost of the circuit

better_than(*other*, *goal*)

Determine whether this instance approximates a goal tech better than some other *Technology* instance.

Parameters

- **other** – another technology to compare against
- **goal** – the goal to compare both against

combined_with(*other*)

Combine with another *Technology* instance and return the resulting, new tech. Note that this operation is NOT commutative. In other words, *a.combined_with(b)* does NOT do the same thing as *b.combined_with(a)*, even if you set the PRNG seed.

Parameters **other** – another technology with which to combine

distance_from(*goal*)

Distance is the number of possible inputs for which the output of the technology circuit differs from that of the goal technology. Smaller values indicate technologies that are closer in functionality.

Basically, this is the number of identical rows in the truth tables of the respective truth tables.

Note that, for now, we compare right-to-left and ignore extra bits if one bit string is longer than the other. So, for example, a circuit that outputs 1101 for a given set of inputs will be considered equal to another circuit for that set of values if the other circuit evaluates to 101.

Todo

Evaluate how to compare bit strings of unequal lengths.

Parameters **goal** – the goal technology to compare against

inputs()
Return a Python set of the inputs of the circuit.

satisfies(*goal*)
Determine whether this instances exactly implements a goal tech.

Parameters **goal** – the goal to compare against

1.2 community

Primary building blocks for simulations based on the circuit model.

class circuits.community.**Community**(*goals=None*, *technologies=None*, *primitives=None*)
A single innovation community. Within the community, technology sharing occurs freely and regularly. New technologies can be composed from old technologies as the community attempts to achieve its goals. Serves the following purposes:

•tracks discovered technologies

•maintains a list of goals

add_technology(*tech*)
If *tech* is superior to an existing technology at satisfying a goal, replace and return the existing technology. Return *None* otherwise.

already_known(*tech*)
Return *True* if *tech* has already been discovered, *False* otherwise.

attempt_invention()
Attempt to create a new technology, return the technology if successful, *None* otherwise.

Attempts to create a new technology through combination of existing technologies, primitives, and constant boolean values.

satisfies_goal(*tech*)
Return the goal best (most closely) satisfied by *tech*.

1.3 variables

Shared set of boolean variables for use in all technologies. Two special constants are exported.

X is the vector of BDD variables that are available for use in all circuits. These can be used, once imported, by indexing *X* like a list: `X[0]`, etc.

N is the length of the *X* vector.

circuits.variables.**evaluated_at**(*circuit*, *bitstring*)
Evaluates the given circuit (function array) at the given bitstring and returns the resulting bitstring. The input and output bitstrings will be lists of *0* and *1*. If the input bitstring is shorter than *N*, it will be padded to the right with zeroes.

Parameters

- **circuit** – the boolean circuit to evaluate
- **bitstring** – a list of *0* and *1* integers

1.4 bitstring

Utilities for dealing with bit strings.

circuits.bitstring.**bitrange**(*length*)

Produces all possible bit strings up to the given length.

Parameters **length** – the maximum length bit string to produce

```
>>> list(bitrange(1))
[[0], [1]]
>>> list(bitrange(2))
[[0, 0], [0, 1], [1, 0], [1, 1]]
```

circuits.bitstring.**increment**(*bs*)

Increments the given bit string.

Parameters **bs** – a bit string (list of integers)

```
>>> increment([1, 0, 1])
[1, 1, 0]
>>> increment([0, 0, 0])
[0, 0, 1]
>>> increment([1, 1, 1])
[1, 0, 0, 0]
```

1.5 primitives

Primitive logic circuit technologies.

The follow primitive technologies are available here:

- *NAND*
- *AND*
- *OR*
- *XOR*
- *ONE* (binary 1)
- *ZERO* (binary 0)

1.6 constants

Simulation constants. Some of these will move to configuration files or command line arguments at some point.

CHAPTER 2

Todo List

Todo

Evaluate how to compare bit strings of unequal lengths.

(The *original entry* is located in /var/build/user_builds/evolutionary-circuits/checkouts/latest/circuits/technology.py:docstring of circuits.technology.Technology.distance_from, line 14.)

CHAPTER 3

Indices and tables

- *genindex*
- *modindex*
- *search*

C