Mathematical Logic and NUMBAS

Available at github.com/Tandethsquire/LogicNode

Andrew Iskauskas

EAMS 2018, Newcastle

August 29th, 2018

UNIVERSITY OF LEEDS

Motivation

Many problems in an introductory course to Logic lend themselves to some degree of randomisation in assessments:

- Validity of Syllogisms
- Conversion of statements to/from Polish notation
- Model equivalence and statements in propositional logic
- Normal forms (conjunctive and disjunctive)

However, implementing such questions in ‘vanilla’ NUMBAS is code-heavy with a lot of repetition between disparate problems.

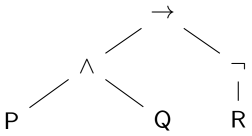
Example: Valuing Statements

One such problem is finding the truth table for $(P \wedge Q) \rightarrow \neg R$. To evaluate this type of statement, we can use the equivalence $P \rightarrow Q \equiv (\neg P) \vee Q$. But when the operands of $\rightarrow$ are complex, converting this becomes a non-trivial exercise in regular expressions, and is extremely sensitive to the presentation of the expression: will $(((P) \wedge (Q)) \rightarrow (\neg(R)))$ give the same result?

Parse Trees

The key ingredient to dealing with many such problems is the use of a *parse tree*:

$$(P \wedge Q) \to \neg R$$



An implementation of this in NUMBAS would allow for valuations of the tree, comparisons of different trees, and a way of obtaining equivalent expressions of the logical statement.

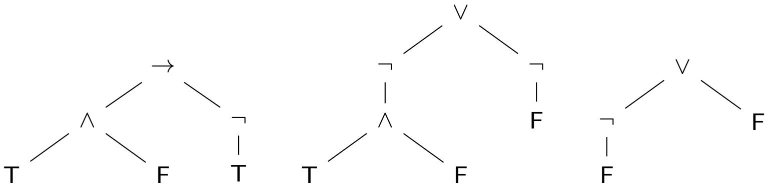
Implementation

In Logic.js, we treat each element of the parse tree as a *node* object, with parent, children, and value properties. By collecting and linking together many such nodes, the parse tree is abstractly represented.

From this, we can extract the logical statement in different notations (particularly Polish and reverse Polish notation), and by giving the variables $P,Q,R\dots$ true/false valuations and *collapsing* the tree, extract the truth table for a statement.

Collapsing

Consider the valuation $P = T$, $Q = F$, $R = T$ of our statement:

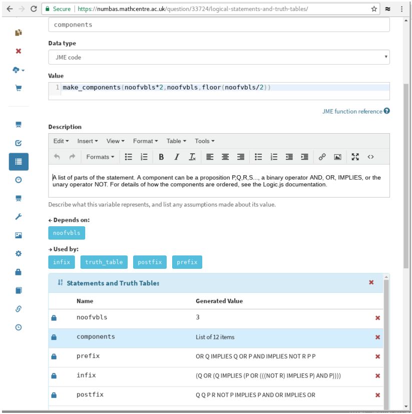


Substitution of → is achieved by modifying the node structure, and collapsing is performed recursively from the bottom up.

Applications

- Expressions: `make_components` produces a collection of nodes; `string_from_tree` converts the nodes into Polish (prefix), reverse Polish (postfix), or standard (infix) notation.
- Truth Tables: `truth_table` creates an array of all possible valuations of the statement
- Model equivalence: `make_model` creates a collections of statements, whose truth tables can be compared to check equivalence
- Conjunctive and Disjunctive Forms: `normal_form` creates (or converts) a tree as ‘an AND of ORs’, or an ‘OR of ANDs’.

Example



https://numbas.mathcentre.ac.uk/question/33724/logical-statements-and-truth-tables/

components

Data type

JME code

Value

```
make_components(noofvbls*2,noofvbls,floor(noofvbls/2))
```

JME function reference

Description

A list of parts of the statement. A component can be a proposition P,Q,R,S..., a binary operator AND, OR, IMPLIES, or the unary operator NOT. For details of how the components are ordered, see the Logic.js documentation.

Describe what this variable represents, and list any assumptions made about its value.

Depends on:

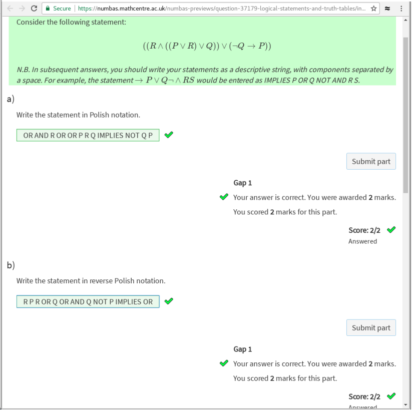
noofvbls

Used by:

infix truth_table postfix prefix

Statements and Truth Tables

Name	Generated Value
noofvbls	3
components	List of 12 items
prefix	OR Q IMPLIES Q OR P AND IMPLIES NOT R P P
infix	(Q OR (Q IMPLIES (P OR (((NOT R) IMPLIES P) AND P))))
postfix	Q Q P R NOT P IMPLIES P AND OR IMPLIES OR



https://numbas.mathcentre.ac.uk/numbas-previews/question-37179-logical-statements-and-truth-tables/in...

Consider the following statement:

$$((R \wedge ((P \vee R) \vee Q)) \vee (\neg Q \to P))$$

N.B. In subsequent answers, you should write your statements as a descriptive string, with components separated by a space. For example, the statement $\to P \vee Q \neg \wedge RS$ *would be entered as IMPLIES P OR Q NOT AND R S.*

a)

Write the statement in Polish notation.

OR AND R OR OR P R Q IMPLIES NOT Q P ✓

Submit part

Gap 1

✓ Your answer is correct. You were awarded **2** marks.

You scored **2** marks for this part.

Score: 2/2 ✓

Answered

b)

Write the statement in reverse Polish notation.

R P R OR Q OR AND Q NOT P IMPLIES OR ✓

Submit part

Gap 1

✓ Your answer is correct. You were awarded **2** marks.

You scored **2** marks for this part.

Score: 2/2 ✓

Answered

Further Applications

Not complete by any means: examples of further work include

- Visualisation of the parse tree (an attempt in this direction can be found in `tree_to_canvas` in LogicNode.js);
- Generating arguments in 'natural' English corresponding to a propositional logic system (an extension of the model generation);
- Complete Operator Sets: given a set of connectives, can the set $\{\neg, \wedge\}$ be expressed?

Join in!

The .js file for the Logic extension can be found at `github.com/Tandethsquire/LogicNode`; pull requests are warmly received.

An example of what can be done with the current functions can be found in the NUMBAS demo exam `numbas.mathcentre.ac.uk/exam/7728/logic-node-demo/`