

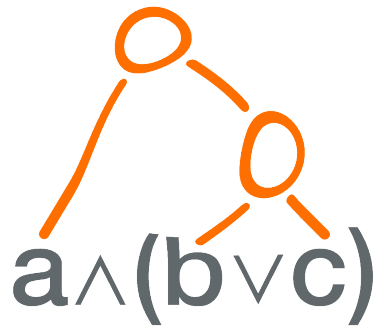


Bachelor Thesis

Nyāya for iPad

Interactive Environment for BoolTool

Alexander Maringele (8517725)
alexander.maringele@gmail.com



9 September 2012

Supervisor: Dr. Georg Moser

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt durch meine eigenhändige Unterschrift, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Alle Stellen, die wörtlich oder inhaltlich den angegebenen Quellen entnommen wurden, sind als solche kenntlich gemacht.

Ich erkläre mich mit der Archivierung der vorliegenden Bachelorarbeit einverstanden.

Datum

Unterschrift

Abstract

If wishes were horses
Beggars would ride:
If turnips were watches
I would wear one by my side.
And if if's and an's were pots and pans,
The tinker would never work!

This nursery rhyme from the 16th century has some propositions! But are they true? Propositional logic provides a toolset to answer this question and many other questions, where propositions are made and connected. **Nyāya for iPad** will introduce the formalism of propositional logic interactively and step by step to the user, so they can translate the poem into a formula of propositional logic. Then they will know, why the poem is true.

Contents

1. Introduction and Motivation	1
1.1. BoolTool	1
2. Related Work	2
2.1. Websites	2
2.2. Apps for iPad	2
3. Processes and Tools	4
3.1. Project phases	4
3.2. Tools	4
3.3. Development Model	4
4. Concept	5
4.1. Tutorials	5
4.1.1. Introduction	5
4.1.2. Syntax	7
4.1.3. Semantics	8
4.1.4. Normal Forms	8
4.1.5. Binary Decision Diagrams	9
4.2. Playgrounds	9
4.2.1. Create formulas by syntax tree	9
4.2.2. Perform equivalence transformations on the syntax tree	9
4.3. Glossary	9
4.4. BoolTool	10
4.4.1. Input	10
4.4.2. Output	10
5. Implementation	11
6. Retrospective and Outlook	12
6.1. Retrospective	12
6.2. Outlook	12
7. Summary and Conclusion	13
List of Figures	15
List of Tables	15
Glossary	17

Bibliography	18
A. Handbook	19

1. Introduction and Motivation


1.1. BoolTool

2. Related Work

Of course, Nyāya for iPad is not the first program, that processes expressions of propositional logic. Besides powerful SAT-solvers and automatic provers, which rarely can be used by beginners, there are some applications, than can be used on a basic level. Most of them provide a (short) introduction to propositional logic, but they seldom offer tutorials or exercises.


The following overview focuses on applications accessible per website and apps available for iPad.

2.1. Websites

- **PLogic Applet** by S. Lukins, A. Levicki, and J. Burg at the Wake Forest University, is a tutorial program for propositional logic “that serves a double role as an educational tool and a research environment”¹ and focuses on interactive theorem-proving.
-  **BoolTool** by the Computational Logic research group at the University of Innsbruck “is an interface to the program BoolTool which allows the manipulation and evaluation of boolean functions.”². It computes truth tables and binary decision diagrams and checks for satisfiability and validity.

2.2. Apps for iPad




Some apps for iPad or iPod touch handle aspects of propositional logic.

-  **Constraints** by Davide Cucciniello is a “SAT based propositional (boolean) logic engine defined by a list of models, where a model includes a list of constraints (a Knowledge Base) each defined by a propositional formula (x and (y or not z)) including a set of propositional (boolean) variables (x,y,z) and operators (and,or,not).”³ let’s you build models in an interactive way. It also provides a short introduction in propositional logic, but no tutorials.

¹ <http://www.cs.wfu.edu/~burg/papers/PropLogic.pdf>

² <http://cl-informatik.uibk.ac.at/software/booltool/?page=info>

³ <http://www.mysvc.it/myapps/constraints/>

-  **Truth Table Generator** by Mertz Werkz LLP “constructs truth tables for the Boolean expressions you enter”.⁴ It does that nicely by using standard symbols of propositional logic (and a limited set of atoms), but nothing more.
-  **Boolean Logic Cheat Sheet** by Clint Johnson presents two pages of logic “rules and laws” in low picture quality. So “all of your Boolean logic needs.”⁵ does not feel quite right.
-  **Logic Mania** by Imagination Creations presents seven logic gates in a list. Their semantics are shown in a per-gate-simulation where the user chooses the input and the app shows the result. A combination of gates is not possible. It’s quite nice but the statement “THE [sic] reference application for students studying logic gates.”⁶ seems exaggerated.

⁴ <http://www.mertzwerkz.com/truth.html>

⁵ <http://itunes.apple.com/at/app/boolean-logic-cheat-sheet/id341959531?mt=8>

⁶ <http://itunes.apple.com/at/app/logic-mania/id434019152?mt=8>

3. Processes and Tools

3.1. Project phases

3.2. Tools

3.3. Development Model

4. Concept

The aim of this project was to design an interactive environment on a tablet computer that allows the user to learn simple facts about the formalism of propositional logic and standard transformations of Boolean functions. The environment is meant to be self-explanatory.

The core of this concept builds on tutorials with exercises and a playground where the learned knowledge can be used in different modes. A glossary of terms and a formula calculator (BoolTool) to check formulas will support the user in their learning.

4.1. Tutorials

Nyāya for iPad covers syntax and semantics of propositional logic, but not natural deduction. It introduces normal forms and the representation of boolean functions as truth tables and binary decision diagrams.

The structure of the tutorials and the content of the exercises are heavily based on the according sections of the book *Logic in Computer Science* [1] by M. Huth and M. Ryan. Additional content is taken from the lecture *Logic* [2] by A. Middeldorp

Every tutorial includes a teaching part with examples (simple html files in utf8 encoding) and interactive challenges, where the user gets immediate feedback. Every interactive part will present some instructions to master the challenge.

4.1.1. Introduction

The first three tutorials give an overview to modeling, i.e. the translation from sentences of natural language to more formal sentences. The content and configuration of the comprehensive questions are stored in xml-files in utf8-encoding.

The content is written into a simple property list “QA124.plist” (table 4.1.1 on the following page) with an array (starts at line 2) of arrays that include sentences and sample solutions (the first exercise-array starts at line 3). The first string is the sample sentence (line 4), and the following entries are the solutions for different exercises.

The tutorials has their own configuration files “NyTuTester101.plist” (table 4.1.1 on the next page), “NyTuTester111.plist” and “NyTuTester121.plist”, with a dictionary that describes the task and defines the index of the sample solution.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.
   apple.com/DTDs/PropertyList-1.0.dtd">
3 <plist version="1.0">
4   <array>
5     <array>
6       <string>If the barmoter falls, then it will rain or it will
          snow.</string>
7       <string>if p then q or r</string>
8       <string>the barometer falls; it will rains; it will snow</
          string>
9       <string>p  $\rightarrow$  q  $\vee$  r</string>
10    </array>
11    <array>
12      <string>sample sentence ...
13      □
14    □
15  □
16 </plist>

```

Table 4.1.: Content file with sentences and solutions

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.
   apple.com/DTDs/PropertyList-1.0.dtd">
3 <plist version="1.0">
4 <dict>
5   <key>keyLabelText</key>
6   <string>Guess the structure of composite sentences</string>
7   <key>inputLabelText</key>
8   <string>Use only “”if, “”then, “”and, ““or, “”not, “”p, “”q, “”r
      .</string>
9   <key>valueLabelText</key>
10  <string>Sample solution (you may have found an other correct
      epression)</string>
11  <key>questionsFile</key>
12  <string>QA123</string>
13  <key>answerIndex</key>
14  <integer>1</integer>
15 </dict>

```

Table 4.2.: Configuration file with comprehension question

Motivation – the composite structure of sentences

The aim of logic in general is to reason about situations.[1] The common structure of two composite sentences in natural language is reduced to “If p and not q , then r . Not r . p . Therefore, q “. If the reasoning is correct, the user can use it in both situations. In the interactive part the user has to guess the structure of composite sentences.

Propositions – declarative sentences

The concept of propositions – declarative sentences – as indivisible building blocks of propositional logic is explained in more detail. Some counterexamples are presented and the ambiguity of natural language is demonstrated. The user has to find the propositions in composite sentences.

Symbols – emphasize the structure

The symbols to replace “if then”, “not”, “and” and “or” are introduced. Now the user should transform the sentences from the first tutorial into pure propositional formulas. The content will be generated by the app and the user’s solution will be checked without tolerance.

Limits

Not every situation can be transformed into a formula in propositional logic. This tutorial has no interactive part.

4.1.2. Syntax

After the informal introduction into the atoms and connectives of propositional logic, the formal aspects of propositional logic will be introduced to the user.

Well formed formulas

The concept of well formed formulas is explained. The user has to build simple well formed formulas: “a conjunction”, “an disjunction of an negation”, “a negation of an implication” and so on. They have to use parenthesis.

Conventions for your convenience

Precedence of connective is introduced. The user has to rewrite formulas to save some parenthesis.

Subformulas

Subformulas (starting from atoms) are used to build bigger formulas. Big formulas are built from many sub formulas. The user has to find all sub formulas of simple composite formulas.

Syntax Trees

This tutorial explains how to draw a syntax tree (starting with atoms) The user has to draw the syntax tree of simple formulas.

Additional symbols – Top and Bottom

The tutorial introduces top and bottom as abbreviations of conjunctions and disjunctions of formulas with their negation (tautologies and contradictions)

Simplifying formulas

The tutorial teaches the simplifying of conjunctions and disjunctions with top and bottom.

4.1.3. Semantics

After the introduction of well formed formulas the meaning of logical connectives is defined.

Truth assignments and valuation of formulas

The user has to guess or calculate the evaluation of a given formula with a given truth assignment of the atoms of the formula.

Truth tables

The user has to fill in complete truth tables.

Entailment and Equivalence

Satisfiability, Tautologies and Contradictions

content is missing

4.1.4. Normal Forms

Implication Free Form

content is missing

Negation Normal Form

content is missing

Conjunctive Normal Form

content is missing

Disjunctive Normal Form

content is missing

Algorithms combined

content is missing

4.1.5. Binary Decision Diagrams

content is missing

4.2. Playgrounds

content is missing

4.2.1. Create formulas by syntax tree

content is missing

Find sub formulas by syntax tree

content is missing

4.2.2. Perform equivalence transformations on the syntax tree

content is missing

Substitute implications

content is missing

Distribute negation over con- and disjunctions

content is missing

Remove double negations

content is missing

Distribute conjunctions over disjunctions and vice versa

content is missing

Substitute Contradictions and Tautologies

content is missing

4.3. Glossary

content is missing

4.4. BoolTool

Of course the interactive environment should contain the functionality of the of BoolTool. Some additions should be added for comfort.

4.4.1. Input

As minimum the embedded BoolTool must support the small latin letters as identifiers for atoms, “T” and “F” as names for top and bottom, and the symbols “!&|+>~” as connectives, because these symbols can be found on almost every keyboard. In environments with native utf8-strings it is desirable that standard symbols of propositional logic “ $\neg \vee \wedge \rightarrow$ ” are accepted too, so the user can copy the output and paste it into the input. This enables the local persisting of formulas in standard representation of boolean functions.

$\top \perp \neg ! \wedge \& \backslash \backslash \backslash + \vee | \oplus | ^ = | < > | \leftrightarrow | > | \rightarrow | \models | () , | ; | \backslash w +$

Table 4.3.: Regular expression for a lexer to build a more flexible parser

Name	NOT	AND	OR	IMPLIES
ASCII	!	& .	+	>
Symbol	\neg	\wedge	\vee	\rightarrow
UTF-8	C2 AC	E2 88 A7	E2 88 A8	E2 86 92
Unicode	U+00AC	U+2227	U+2228	U+2192
HTML	¬	∧	∨	→

Table 4.4.: basic connectives in ascii, utf-8, unicode and html

TOP	BOTTOM	XOR	EOR	IFF
T 1	F 0	\wedge	\oplus	$< >$
\top	\perp	\vee	\oplus	\leftrightarrow
E2 8A A4	E2 8A A5	E2 8A BB	E2 8A 95	E2 86 94
U+22A4	U+22A5	U+22BB	U+2295	U+2194
T	F		⊕	↔

Table 4.5.: additional connectives in ascii, utf-8, unicode and html

4.4.2. Output

BoolToll calculates and displays a bundle of representations of user’s input.

5. Implementation

6. Retrospective and Outlook

6.1. Retrospective

6.2. Outlook

7. Summary and Conclusion

List of Figures

List of Tables

4.1. Content file with sentences and solutions	6
4.2. Configuration file with comprehension question	6
4.3. Regular expression for a lexxer to build a more flexible parser . .	10
4.4. basic connectives in ascii, utf-8, unicode and html	10
4.5. additional connectives in ascii, utf-8, unicode and html	10

Glossary

computer is a programmable machine that receives input, stores and manipulates data, and provides output in a useful format.

naïve is a French loanword (adjective, form of naïf) indicating having or showing a lack of experience, understanding or sophistication.

Bibliography

- [1] M. Huth and M. Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems*. Cambridge University Press, New York, NY, USA, 2004.
- [2] A. Middeldorp. Logic (lecture). <http://cl-informatik.uibk.ac.at/teaching/ws11/lics/content.php>, 2011/2012.

A. Handbook