Welcome to the final presentation of my bachelor project Nyaya for iPad,
an interactive environment including BoolTool.
I am Alexander Maringele,
and my supervisor is Dr. Georg Moser.

Nyāya – origin and meaning

First of all, I want to explain to you the name Nyaya and its origin.
Nyaya is sanskrit and its literal meaning is recursion
in the sense of syllogism or inference.
Nyaya is one of the orthodox schools of Hindu philosophy,
specifically the school of logic.
One of its central tenets is
"that obtaining valid knowledge is the only way to obtain release from suffering."
And valid knowledge, on the other hand,
is also an important goal of many studies, especially for the subject of logic.

Logic – motivation

In their book "Logic in Computer Science", Huth and Ryan write
"The aim of logic in computer science is
to develop languages to model the situations
we encounter as computer science professionals,
in such a way that we can reason about them formally."
Let's take, for example, the beginning of the famous nursery rhyme
"If wishes were horses, beggars would ride."
We could ask ourselves whether or not this statement is true.
So we split it into two parts, first the condition and second the result,
which leaves us with a model "if-then".
This can be formalized into "p implies q".
Now we can calculate the valuation,
and we realize that the valuation of "p" must be false,
because wishes are not horses in our domain.
Therefore, following the principle that anything
can be concluded from an untrue condition, the complete statement is true.

CL BoolTool

Next, I want to give you a short introduction to BoolTool
created by the computational logic research group at the university of innsbruck
because its functionality is included in my project.
BoolTool is a powerful means for manipulating and evaluating formulas in propositional logic.

It defines an input syntax for formulas
It derives normal forms from formulas
It computes truth tables and binary decision diagrams
and it calculates satisfiability, tautologies and contradictions.

Unfortunately, it is not really suitable for beginners because up to now
It does not motivate or explain much,
It does not use standard symbols of propositional logic,
It does not explain equivalence transformations,
and it does not define normal forms.

Aim

The aim of my project now is to provide
an interactive learning environment for propositional logic.
It should allow the user to learn the formalism of propositional logic,
it should clarify the separation of syntax and semantics.
It should introduce normal forms like negation normal forms, conjunctive and disjunctive normal forms to the user.
It also has to demonstrate standard transformations of Boolean functions and the coherence of different representations of a Boolean function.

All this should be supported by a self-explanatory environment.

Concept

To reach this goal, I developed a platform agnostic concept
which supports the most effective learning techniques
– steady learning and practice testing –
by combining small bits of learning content with literally countless exercises.

Nyaya offers short tutorials for general concepts and definitions as well as
exercises to consolidate the learned concepts and definitions.
It also provides a playground for building and transforming syntax trees
and a glossary of technical terms.
It includes an user interface that works similar to the web interface of BoolTool.

Demo – Tutorials

Let me demonstrate to you now this concept in its platform specific implementation for iPad. Tapping on the symbols at the bottom of the screen you can switch between pages of Nyaya.

The second page gives you a table of the tutorials as well as the tutorials themselves. After choosing a topic you can either read the tutorial or go directly to the exercises and tests provided.
When you choose the topic syntax trees, for example, you will find such a tree and will be asked to transform it into a formula. Does anyone want to have a go? After writing your suggestion, you push the check button to find out if your solution is correct or not. When your done with the exercises, you push the close button to return to the tutorial.

Demo – Playground

The playground on page three, on the other hand, is an area for experimentation. Here you can create syntax trees and manipulate them. You can add a syntax tree to the playground by tapping a formula in the list of formulas given. For creating a new simple tree yourself, just tap and hold a free spot on the screen. You select a syntax tree by tapping it. This page is called a playground because by tapping around you find out about all the functions provided. There is also a hint button, which helps you along if you get stuck. There are two modes in which you can experiment along. The locked restricts you to equivalence transformations, whereas the unlocked mode even allows semantical changes. As you can see, equivalence transformations usually lead to bigger formulas. A tap on the red symbol will close your syntax tree.

Demo – BoolTool

Last but not least, I want to present to you Nyaya's BoolTool. After typing a formula, you tap on the evaluation button to get normal forms, a truth table and a binary decision diagram for your formula. Additionally, you are provided with information about satisfiability and validity of your formula.
This was the demonstration. Let's continue with the slides.

Platform

Now, what were my motivations for choosing iPad as the platform for my application. The most popular computing devices nowadays are notebooks, phones and tablets. As the tablet allows you a complete immersion into one program by blending out all other applications, it seems to me the perfect device for an application featuring a learning environment.
HTML5 being too time-consuming for developing an appealing application including interactive graphics, which fits for all browsers, screen sizes, and input devices, I limited my pool of development environments to Android or iPad. I decided for iPad because I own one and thus I can run my application on the device itself. Moreover, iPad has popularized the use of tablets. More than a hundred and fifty million iPads have been sold up to now.

Toolchain

For developing apps for iPad, you need a Mac, and you have to register at the Apple Developer Portal. Then you can download your developer IDs and Certificates, along with Xcode, the integrated development environment for Mac OS and iOS. You have to learn Objective-C and Cocoa Touch, the language and framework used by Xcode. Xcode includes Git as a version control by default, but you can use subversion repositories too. Although Xcode includes an iPad simulator, it is highly recommended to proof-run your application on a genuine device. I would also recommend the use of GitHub or a similar service for backup.

Execution - Fail Fast

My project execution did not follow a specific development model, but I used some principles of agile development with scrum instead. One of its major principles is to fail fast. This encourages you to find critical points of your project as soon as possible, and to decide in a given amount of time, if the problem can be overcome. If not, you have to find an alternative to your original plan. This actually happened when I wanted to use the OCaml source code of BoolTool for my application. After several failing attempts, I decided to rewrite the functionality of BoolTool in Objetive-C.

Execution - Cont.

I also created a lot of tests before its implementation,
which I used to prove the correct running of the application
at any time of the project process. This principle is called test driven development.
While implementing, I rewrote classes and methods to improve the quality of the code,
This refactoring was always secured by unit-tests written before the first implementation.
For example I switched from mutable to immutable data structureswithout hassle in the midst of the project .
And by writing down a detailed concept, I already defined all use cases of the application.

Commits

I started my project by exploring interface possibilities.
After developing some prototypes, I implemented the core components.
I added content, controllers and configurations to make my application ready for use.
And finally, I reviewed and tested all parts of Nyaya multiple times.
The commit history at GitHub reflects those four phases.
The main work was done in the second phase by developing the core components like parser, views of syntax trees, truth tables, and binary decision diagrams.

Tutorials

Let me give you a short overview of the tutorial contents.
After a section giving an informal introduction to using propositional logic for modeling situations and to its limits, the syntax of propositional logic is defined and demonstrated. Nyaya supports strict syntax with many parentheses as well as a more general syntax using precedences and associativity. The concepts of sub-formulas and syntax trees are introduced, and some basic equivalences are described.
In the semantics section, truth assignments and the valuation of formulas and sub-formulas are introduced. Semantic entailment and equivalence, satisfiability and validity are defined and demonstrated with truth tables.
Normal forms, like conjunctive normal form, are defined, and equivalence transformations to derive normal forms from arbitrary propositional formulas are introduced.
In the last few tutorials Boolean functions are introduced, and the presentation of a Boolean function as a binary decision diagram is explained.
The exercises to all these tutorials make use of Nyaya's random formula generator, which produces bigger formulas after each correct answer.

Playground

As you have seen in the demonstration, Nyaya supports the display of multiple syntax trees. These syntax trees can be manipulated with touch gestures. Truth values can be assigned to the atoms of the syntax tree, and the sub-trees will be evaluated automatically.

BoolTool

Nyaya's reimplementation of BoolTool accepts symbols from propositional logic as well as Boolean expressions, since this produces no ambiguities in the definition of a Boolean function. The output includes validity and satisfiability, negation, conjunctive and disjunctive normal form, a truth table and a reduced ordered binary decision diagram. Truth tables are abbreviated if they have more than 128 rows.

Model

Parser and node class constitute the core of Nyāya.

Nyāya uses a regular expression to transform the input string into a stream of tokens. Special symbols and words are transformed into operator tokens, all other words are recognized as identifiers. Since the cocoa framework works with Unicode-strings, Chinese words can be used as identifiers.

An EBNF grammar for propositional logic, adapted to be in accordance with precedence and associativity used in BoolTool, was implemented into a recursive descendent parser, following the principles outlined in "Compiler Construction" by Kenneth C. Louden.

The parser reads the stream of tokens, uses factory methods of the abstract node class to create all necessary sub-trees, like atoms and conjunctions, and returns the root node of the syntax tree derived from the input.

To enable Nyaya to transform syntax trees, the decorator pattern is used for adding additional methods to the node class and it's private sub-classes.

Graphs

The in-memory representation of syntax trees is optimized.
Nyāya 's syntax trees are designed as immutable reduced acyclic object graphs.
Every syntax operation will create a new acyclic object graph by re-using sub-formulas multiple times.

Although immutable, the "colour" of leaf nodes, the truth assignment, can be changed, because changing the colour does not change the object graph, which directly represents the syntax of the formula.

Retrospective

Nyaya is in a stable and fully functional state but not everything is perfect. To distribute the correction of a typo in a tutorial, a new version of Nyaya has to be submitted to the app store. It usually takes one week from submitting to publishing an app. It would be possible for Nyaya to download content directly from a web server, but this feature has not been added yet.

Although sub-formulas are re-used, the valuation process re-calculates a sub-formula every time it is used. The creation of reduced ordered binary decision diagrams is very slow. You can paste formulas into the input field of bool tool, but you cannot export normal forms.

If your eyesight is bad, "voice over" and "zoom" would be helpful for using the application. If your physical and motoric skills are restricted, "AssistiveTouch" could help you. To my disgrace, I must admit Nyaya still does not support these standard accessibility techniques of iOS (very well.)

Outlook

Nyaya is available in English only. So far, there were downloads of the application in USA, Canada, Austria and Belgium. Translations and localization would help to extend the market. A German version will be released this year.

Some additional features,
like creating and editing binary decision diagrams in a similar way to syntax trees, would be nice.

The use of n-ary Boolean functions in formulas and syntax trees would be an obvious extension to Nyaya . Actually the parser would support it, but Nyaya's syntax tree and BoolTool do not.

Syntax trees, truth tables and BDDs should be exportable, too.

I think, there is a small chance of creating a version for Mac or Android.

Distribution

Nyaya is distributed via AppStore and GitHub.

In the app store, you can buy and download it.
You will receive a promo code to download it for free,
when you write an E-Mail to nyaya@maringele.at
with a reply-address of the University of Innsbruck.
You can also download the source code from GitHub
and run it on a Mac with Xcode and iPad Simulator.
To install Nyaya on your iPad, using the source code,
you will need a developer certificate from Apple.
Or you have jailbreaked your device.

The source code is not intended for re-use or redistribution.

Sources

At last, I want to mention the pillars, my application is built on.

The content is based on "Logic in Computer Science" by Huth and Ryan
and the Lecture “Logic” of professor Middeldorp.
The architecture of Nyāya was guided by
“Design Patterns: Elements of Reusable Object-Oriented Software”
by the Gang of Four.
Nyāya’s Parser was built following principles taken from
“Compiler Construction: Principles and Practice” by Louden.
The platform-specific tasks were mastered with the help of
“Cocoa Design Patterns” by Buck anyd Yacktman
and the online iOS developer documentation by apple.

Thank you for your attention! I am now open to answer all remaining questions.