Translating Logic to Natural Language

- Run Zhang
- Huaijin Ning
- Haoyang Tao

CS 4TB3 - Apr.17, 2022

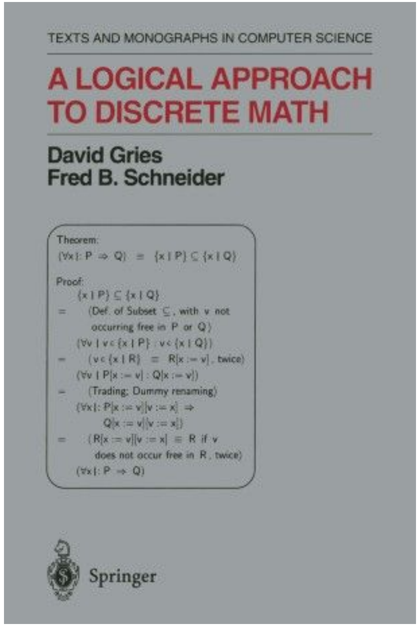
Logic formula

- **Propositional logic**
  - OR (∨)
  - AND (∧)
  - Negation/ NOT (¬)
  - Implication / if-then (⇒)
  - If and only if (≡).
- **Predicate Logic**
  - Universal Quantifier
    ∀x
  - Existential Quantifier
    ∃x

¬ p
p ∨ q
p ∧ q ⇒ r
p ⇒ q ≡ ¬ p ∨ q
…
…
…
((false ⇒ r) ∨ ((¬ z) ∨ h)) ∧ (p ∨ true) ≡ (true ∧ p) ∨ (false ∨ true)
(∀ x ▮ Q ∧ R • P) ≡ (∃ x ▮ (true ∧ (true ∧ false)) • F ∧ (p ∨ r))

CALCCHECK

A Proof-Checker



Research challenge

- List of symbol
  Official meaning/translate for each symbol
- Combination of multiple symbols
  How to understand the meaning correctly?

TABLE 2.3. TRANSLATION OF ENGLISH WORDS

and	becomes	$\wedge$
or	becomes	$\vee$
not	becomes	$\neg$
it is not the case that	becomes	$\neg$
if p then q	becomes	$p \Rightarrow q$

Operator $\vee$ is called *disjunction* or *or*. Expression $b \vee c$ is read as "b or c", because it is *true* iff b or c (or both) is *true*. Operands b and c of $b \vee c$ are called *disjuncts*.

Operator $\wedge$ is called *conjunction* or *and*. Expression $b \wedge c$ is read as "b and c", because it is *true* only if both operands b and c are *true*. Operands b and c of $b \wedge c$ are called *conjuncts*.

Operator $\Rightarrow$ is called *implication*. Expression $b \Rightarrow c$ is read as "b implies c" or as "if b then c". Operands b and c are called the *antecedent* and *consequent*, respectively. Note that $b \Rightarrow c$ is *true* if b is *false*. This is consistent with the usual English interpretation of a statement like "If Schneider is ten feet tall, then Gries can walk on the ceiling" as being *true* simply because Schneider is not ten feet tall. False implies anything, as the saying goes. We discuss implication in more detail in Sec. 2.4.

The symbol $\forall$, which is read as "for all", is called the *universal* quantifier. Expression (9.1) is called a *universal quantification* and is read as "for all x such that R holds, P holds."

The symbol $\exists$, which is read as "there exists", is called the *existential* quantifier. The expression is called an *existential quantification* and is read as "there exists an x in the range R such that P holds". A value $\hat{x}$ for which $(R \wedge P)[x := \hat{x}]$ is valid is called a *witness* for x in $(\exists x \mid R : P)$.

Output format

Single line output:

$p \lor q \land \text{true} \equiv \text{false}$

p or q and true equivalence false

Pointer:

$p \lor q \land \text{true} \equiv \text{false}$

```
p
  ^ or
     ^ q and true
                ^ equivalence
                  ^ false
```

Arrow:

$p \lor q \land \text{true} \equiv \text{false}$

p or ↓
q and true↓
equivalence↓
false

Order number:

$p \lor q \land \text{true} \equiv \text{false}$

p or q (1)
(1) and true (2)
(2) equivalence false

Natural language:

$p \lor q \land \text{true} \equiv \text{false}$

The left hand side in natural language is
calculate p or q
calculate the result of above and True

The right hand side in natural language is False

Finally, the result of LHS is equivalent to the result of RHS.

It's better to make it in multiple line!

Need a true/false evaluator?

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \equiv Q$
False	False	True	False	False	True	True
False	True	True	False	True	True	False
True	False	False	False	True	False	False
True	True	False	True	True	True	True

Truth tables for five logical Connectives

```
#print(evaluate('false ∧ true ∧ (p ∧ true)'))
# ast('false ∧ q ∧ (p ∧ true)')
# evaluate('false v true v (p ∧ true)')
# evaluate('false ∧ true ∧ (true ∧ true)')
print(evaluate('(p v (true ∧ (false v true))) ≡ (true ∧ true)'))
# False and True equal to False ↓
# false ∧ (true ∧ true)
# false ∧ true
# """
# False and True equal to False ↓

#                                            True and True equal to True ↓

#                               'False and True'
# """
```

```
(p v (true ∧ (false v true))) ≡ (true ∧ true)
                      ^  True ↓
                                  ^  True ↓
p or True and True is equivalent to True
```

We focus more on the translation in natural language! :D

Priority (logic sequence)

- Priority of input string is implicit, so parentheses!
- And (⋀) / Or (⋁)
- Nested identical symbols
  - Equivalence
  - Imply
- How to use natural language

  to demonstrate priority?

Operator	Precedence
¬	1
∧	2
∨	3
⇒	4
≡	5

Translation example: p ∧ (q ∨ r)

"p and q or r" ?

"p and (q or r)" ?

"q or r ↓"
"p and q or r" ?

"1. q or r"
"2. p and q or r" ?

"q or r"
"p and the result of above" ?

Translation Scheme

Predicate logic

∀ x ❙ Q • P	for all x, if step: R returns true, then T.
∃ x ❙ Q • T	exist an x that matches the following statements Statement: R, Statement: T.

Propositional logic

$\neg$ q	the negation of q
p $\wedge$ q	calculate p and q
p $\vee$ q	calculate p or q
p $\Rightarrow$ q	calculate p implies q
p $\equiv$ q	The left hand side in natural language is p The right hand side in natural language is q Finally, the result of LHS is equivalent to the result of RHS.

Deviation

- Division of work.

  Spent more time working together instead of dividing tasks and working separately.

- Evaluator.

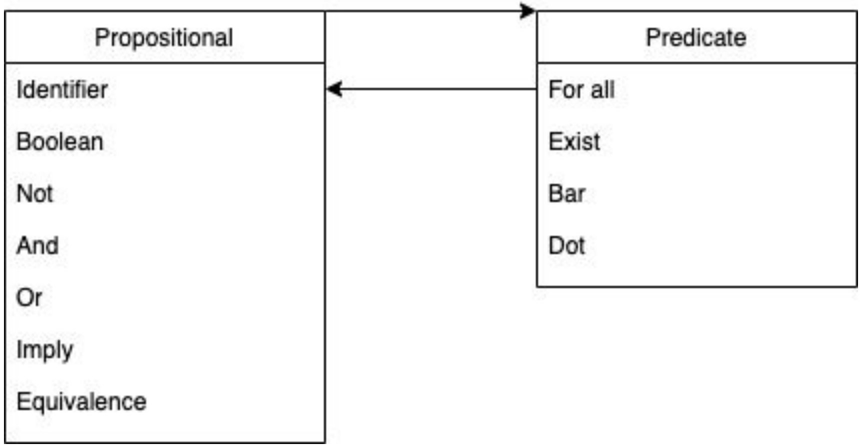
  Had two plans for the evaluator and successfully implemented one, but finally decided not to use it.

Project Structure

Statistics:
Scanner: 64 lines
Propositional: 164 lines
Predicate: 37 lines
Procedure: 61 lines
Testcase: 21 lines

Testcase: 50 cases

- Five notebooks
  - Scanner
  - Propositional logic
  - Predicate logic
  - Grammar procedures
  - Test cases
- Logic classes



Conclusion

- So Far
  - Translator functions
  - Insightful to work with
  - Understand more about the interpreter of recursive definitions

- Future Step
  - Add more logic in predicate part
  - Improve the translation and fix errors for the evaluator
  - Add a set of logically related vocabulary to substitute expression

Example:

$p \Rightarrow q$

- p: It rains
- q: Jake won't walk to school

Then, use logic imply to make our statement "If it rains, then Jake won't walk to school."

Demo

Jupyter Notebook is used for interactive demo.

References

1. Textbook by David Gries and Fred B. Schneider: A Logical Approach to Discrete Math
2. Propositional Logic in Discrete Math: https://ggc-discrete-math.github.io/logic.html
3. Propositional Logic Syntax: https://www.logicthrupython.org/chapter01.pdf
4. Getting Started with the CalcCheck Language: http://calccheck.mcmaster.ca/CalcCheckDoc/GettingStartedWithCalcCheck.html
5. Getting Started with CalcCheckWeb: http://calccheck.mcmaster.ca/CalcCheckDoc/GettingStartedWithCalcCheckWeb.html
6. CalcCheck Syntax: http://calccheck.mcmaster.ca/CalcCheckDoc/2019-12-20_CalcCheck-Syntax-Hints.pdf
7. CalcCheck Theorem List: https://www.cas.mcmaster.ca/~kahl/CS2DM3/2018/2DM3-2018-Mid-November-ThmReference.html