実務に役立つかもしれない数理論理学

自己紹介

フロントエンドチームの金野です。
最近はReact/TypeScript書いたりしています。

食べログ フロントエンドエンジニアブログにも色々記事を書いています。
https://note.com/tabelog_frontend

2020年1月からアメリカの通信制大学に入ってコンピューターサイエンスを学び直し中です。

今日のテーマ

今日のテーマはフロントエンドには1ミリも関係なく、数学の話です。

めちゃくちゃ数学には苦手意識があったので最近数学を学び直し中です。
（高校数学／離散数学／線形代数あたり）

ちゃんとComputer Scienceや数学を学んでる方も多いと思うので~~ボコボコにされにきました~~
超基礎的な内容ばかりですが、間違いや正確さに書ける表現は優しくフィードバックいただけたら嬉しいです。

数理論理学 - Mathematical logic とは

いつも我々が使っている `Boolean` というデータ型。
これは論理推論を代数で展開する「ブール代数」を考案したジョージ・ブール（1815-1864）に由来する。

- 離散数学の一分野だが、数学の基礎知識。
    - 「離散」とは「連続」の反対。「とびとび」になっているものを対象とした数学。
    - 対照的なのが微積分などの解析学で、それらは連続な対象を扱う。
- 科学的思考の基礎となる重要な概念。
- 高校数学では数学I「集合と命題」というトピックで触れる。
- 計算機は0と1の2つの信号を扱うことから、離散数学の守備範囲。離散数学なしに計算機は動かない。

> 参考：イラストで学ぶ　離散数学

普段プログラミングしている私達は論理で飯を食っていると言っても過言ではない（過言）

Vocabulary

真理値 - truth value, logical value

- 真 - trueまたは1
- 偽 - falseまたは0

論理変数 - Logical variable

- 真理値を取る変数。
- PとかQとかRが使われることが多いが、なんでもよい。

ステートメント - Statement

真または偽を表す平叙文（Declarative sentence）

ステートメントは命題論理式(Propositional logical expression)とか単に命題 (Proposition)とも言う場合もある。

（厳密には違うかもしれない。教えて詳しい人）

Atomic statement

Statementのうち、これ以上分割できない文をAtomic statementという

- 月はチーズでできています。（偽）
- 3 + 7 = 10 （真）
- 5 > 6（偽）

Molecular statement

Atomic statementに分割できるものはMolecular statement

- ポチはお行儀よくしていれば、ホネっ子がもらえる（真）
  - P:「ポチがお行儀よくしている」（真）
  - Q:「ホネっ子がもらえる」（真）
  - 論理演算子を使って `P→Q` と書ける
- [JavaScript] `4 > 3 && false` （偽）
  - P: `4 > 3` （真）
  - Q: `false`
  - 論理演算子を使って `P ∧ Q` と書ける

論理演算子 - Logical operator, Logical connectives

論理演算子を使用してMolecular statementを作成することができる。

- `¬`
  - not
  - 否定 - Negation
  - JavaScriptだと`!`

- `∧`
  - and
  - 論理積 - conjunction
  - JavaScriptだと `&&`
- `∨`
  - or
  - 論理和 - disjunction
  - JavaScriptだと `||`

- `⇒` または `→`
  - if ... then ...
  - 包含 - implication, conditional
- `⇔`
  - if and only if ...
  - 双条件 - biconditional

徐々にプログラミングとのつながりが見えてきましたね

論理演算子の優先順

優先度が高い順に

> $\neg$
>
> $\wedge$
>
> $\vee$
>
> $\Rightarrow$ $\Leftrightarrow$

JavaScriptも同じですね。

以下の順で演算子が適用されます。

> 否定 !
> 論理積 &&
> 論理和 ||

https://developer.mozilla.org/ja/docs/Web/JavaScript/Guide/Expressions_and_Operators

なので、`x > 5 || y === 3`を先に評価させるつもりで以下のような条件式を書いたら意図と違った結果になってしまいます。

```
if(x > 5 || y === 3 && isChecked)
```

より優先度の高い演算子`()`を使って上書きしましょう。

```
if((x > 5 || y === 3) && isChecked)
```

真理値表 - Truth tableとは

P	Q	P∨Q	P∧Q	¬P	P⇒Q	¬P∧Q	P⇔Q
0	0	0	0	1	1	0	1
0	1	1	0	1	1	1	0
1	0	1	0	0	0	0	0
1	1	1	1	0	1	0	1

こういう表を真理値表といいます。

基本情報技術者試験でも出てきたような気がします。
複雑な条件式を読んだり書いたりするときに、真理値表を書くと整理しやすいかもしれません。
~~まあconsoleで実際にコードを書いて出力してみるのが一番早いんですけど…~~

ちなみに世の中にはTruth table generatorという便利なツールがあります。

https://web.stanford.edu/class/cs103/tools/truth-table-tool/

論理学のちょっとおもしろい話

`P→Q` がPが真、Qが偽のときだけ結果が偽になり、あとはすべて真になるのにお気づきでしょうか。

Pが偽でも常に結果が真になります。

P	Q	P⇒Q
0	0	1
0	1	1
1	0	0
1	1	1

さきほどのこちら。

- ポチがお行儀よくしていればホネっ子がもらえる（真）
  - P:「ポチがお行儀よくしている」（真）
  - Q:「ホネっ子がもらえる」（真）
  - 論理演算子を使って `P→Q` と書ける

Pを

`「ポチがお行儀よくしていない（つまり行儀が悪い）」（偽）`

としてみたらどうでしょう。

この場合でも、「ポチがお行儀が悪いがホネっ子がもらえる」というステートメントは**「真」**になります。

なぜなら、さきほどの「ポチがお行儀よくしていればホネっ子がもらえる」はPが真（ポチがお行儀よくしている）の場合のことしか決めていないためです。
Pが偽の（お行儀が悪い）場合どうなるかは何も決めていないので、結果はどうあれステートメントには反してないのです。

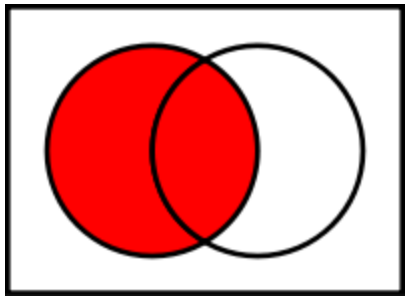
> 参考：イラストで学ぶ　離散数学

つまり、以下の奇妙なステートメントは **論理学的には真になります** 。
前半のifの部分（P)が偽になるのでQに何が書いてあるかは関係がありません。 以下のP→Qの真理値は真です。

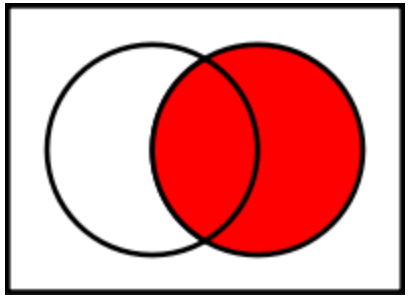
- 地球は円盤なので、海は凍らない（真）
  - P: 地球は円盤である（偽）
  - Q: 海は凍らない（偽）
  - P → Q （真）
- 8が素数ならば、円周率の7624番目は8になる（真）
  - P: 8が素数（偽）
  - Q: 円周率の7624番目は8（？？調べるのが面倒）
  - P → Q （真）
    - 「円周率の7624番目は8」が真であろうと偽であろうとP→Qは真

論理式はベン図でも表現できます

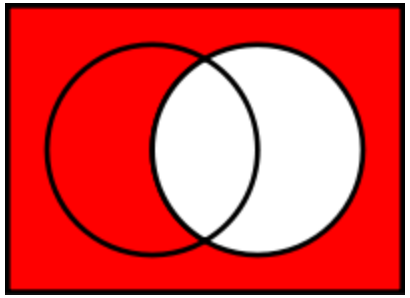
Pがtrue



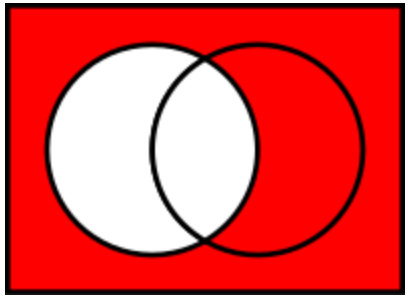
Qがtrue



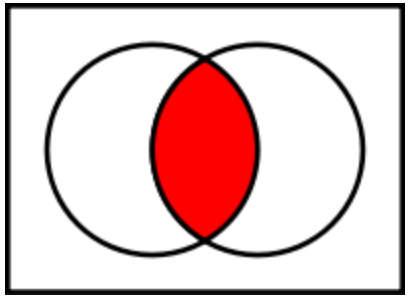
¬Q



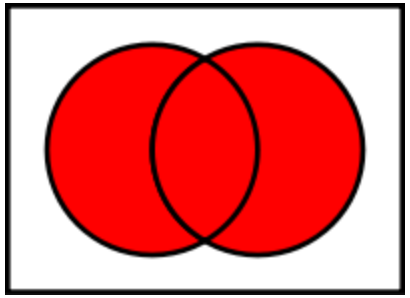
¬P



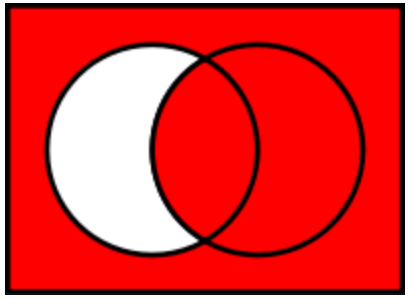
P ∧Q (conjunction)



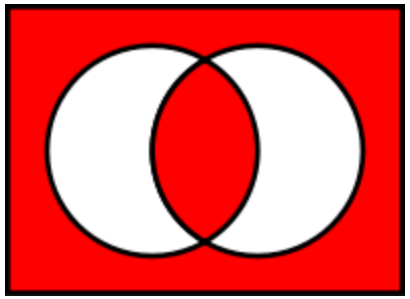
P∨Q (disjunction)



P→Q (implication)



P⇔Q (biconditional)



わけがわからなくなったらベン図を書けば、シンプルな条件文に書き換えられるかもしれないですね。

ドモルガンの法則

- `¬(P∨Q)=(¬P)∧(¬Q)`
- `¬(P∧Q)=(¬P)∨(¬Q)`

これも基本情報技術者試験や高校の数学1で出てきた気がしますね。

真理値表やベン図を書けば、両辺が一致することは簡単にわかります。

¬(P ∨ Q) = (¬P) ∧ (¬Q)

```
if(x !== 5 && !y )
```

↓

```
if(!(x === 5 || y))
```

¬(P∧Q) = (¬P) ∧ (¬Q)

```
if(x !== 5  || !y )
```

↓

```
if(!(x === 5 && y))
```

これを覚えておけば、複雑で分かりづらい条件式も少しリファクタできるかもしれません。

命題論理式における様々な性質

他にも様々な性質があります

> 参考：イラストで学ぶ　離散数学

排中律(excluded-middle law)

- `P ∨ ¬P = 1`
- `P ∧ ¬P = 0`

冪等律 (idempotence law)

- `P ∨ P = P`
- `P ∧ P = P`

交換律(commutative law)

- `P ∨ Q = Q ∨ P`
- `P ∧ Q = Q ∧ P`
- `P ∨ Q = Q ∨ P`
- `P ∧ Q = Q ∧ P`

結合律(associative law)

- `P ∨ (Q ∨ R) = (P ∨ Q) ∨ R`
- `P ∧ (Q ∧ R) = (P ∧ Q) ∧ R`
- `P ∨ (Q ∨ R) = (P ∨ Q) ∨ R`
- `P ∧ (Q ∧ R) = (P ∧ Q) ∧ R`

ここらへんまでは「そりゃそうですね」って感じですね

分配律(distributive law)

- `P ∨ (Q ∧ R) = (P ∨ Q)∧ (P ∨ R)`
- `P ∧ (Q ∨ R) = (P ∧ Q)∨ (P ∧ R)`
- `P ∨ (Q ∧ R) = (P ∨ Q)∧(P ∨ R)`
- `P ∧ (Q ∨ R) = (P ∧ Q)∨(P ∧ R)`

吸収律(absorption law)

- `P ∨ (P ∧ Q) = P`
- `P ∧ (P ∨ Q) = P`
- `P ∨ (P ∧ Q) = P`
- `P ∧ (P ∨ Q) = P`

この分配律と吸収律は覚えておくと役立つかもしれません。

まとめ

- 0と1を扱う論理学は離散数学の基礎
- 論理は科学的思考の基礎となる重要な概念。
- 計算機も0と1の2つの信号を扱うことから、離散数学の守備範囲。
- ステートメントは真または偽を表わす文。
- 論理演算子を使用してmolecular ステートメントを作成できる。
- 真理値表やベン図で論理演算の結果を整理できる
- ド・モルガンの定理など論理式の様々な性質をしっていたら業務でも役立つかもしれない

参考文献

- イラストで学ぶ　離散数学
- 論理演算 - Wikipedia
- Logical connective - Wikipedia
- Mathematical Statements
- Discrete Mathematics - An Open Introduction