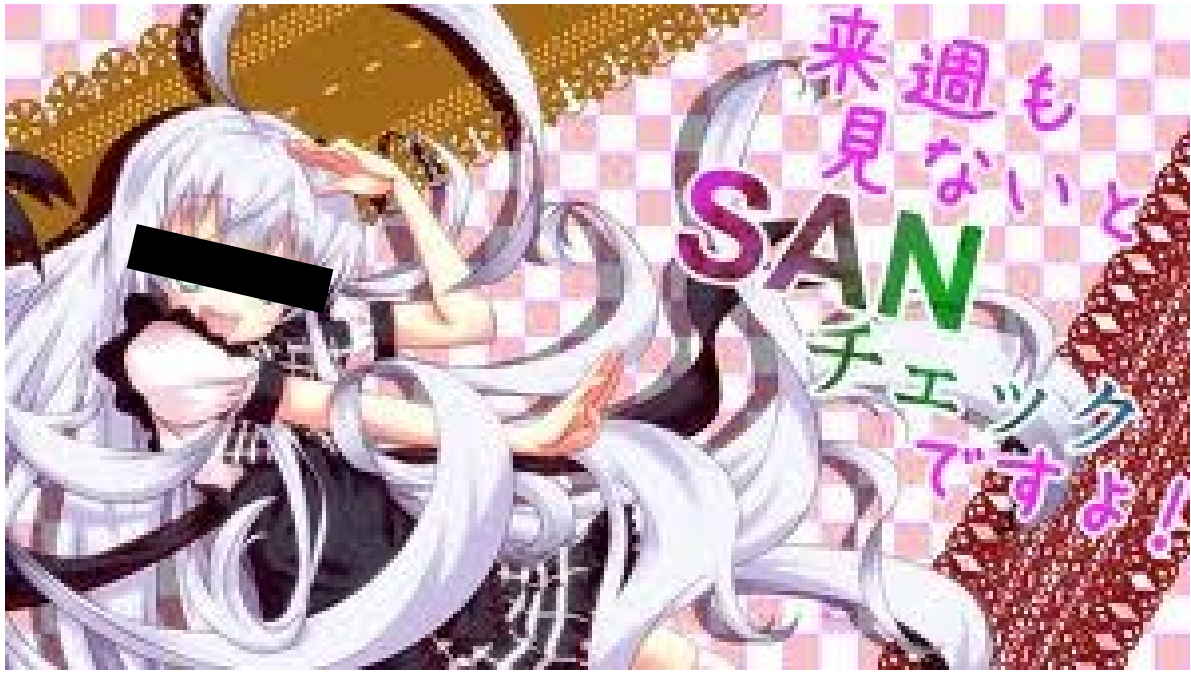

SAN値論理
と
名状しがたい SAN値Prolog

(」･ω･)」うー!(／･ω･)／にゃー!

2012/APR/22, 23
たけおか@たけおかラボ/AXE
@takeoka

二値論理

- 普通の二進数の論理
  - 0/1しか値がない
- 電気回路で扱うのに便利

- 論理型言語の場合、ある文(節)が単純に真か偽で、表せる

三値論理

- 情報理論上では e進数(e:自然対数の底)がもっとも情報の表現効率がいい
  - e=2.718281828
- 2.72進数などという数は、普通、扱いにくいわいっ

  ということで…
- 3進数が、普通に、情報の表現効率がもっともいい
- 三値論理
  - 0/1/2の状態を持ち動作する論理回路を作ってみたり

- 三値論理を論理型言語に取り入れても、得るものはないだろう(と、思う)

いつもニコニコ SAN値論理

- SAN値とは
  - 頭の正気度だにゃー
  - SAN値が0だと、完全な ≠チガイ

- 論理型言語にSAN値というものを入れる
- 評価時の「頭のおかしさ具合」を、「SAN値」と呼ぶ
- 各事実(節)ごとに、「SAN度」を設定できる

- 評価時に
- SAN値に近いSAN度を持つ節が選ばれる

いつもニコニコ SAN値論理

- SAN値に近いSAN度を持つ節が選ばれる
  - SAN値が0 だと、めちゃくちゃに選ぶ(≠チガイだから、仕方ない)
- 節は評価して、真であったら、SAN値が比較される
  - よって、すべての節は評価される
  - 純粋な一階述語論理
  - 順序に依存してはいけない

90:partner(ニャル子).

60:partner(クー子).

30:partner(ハス太).

SAN値が90ぐらいのときに選ばれる

SAN値が60ぐらいのときに選ばれる

SAN値が30ぐらいのときに選ばれる

SAN度

いつもニコニコ SAN値論理

- SAN値に近いSAN度を持つ節が選ばれる
- 論理式もOK

  足し算が、正常にできるか?

  90:plus(*X, *Y, *Z) :- *Z = *X + *Y .

  60:plus(*X, *Y, *Z) :- *Z1 = *X + *Y, ***Z = *Z1 +1** .

  30:plus(*X, *Y, *Z) :- *Z = *X × *Y .

正常

ちょっとおかしい

だいぶおかしい
掛け算してる

SAN度

似た研究: ファジィ論理 Prolog

- 論理型言語にファジィ(確率?)を入れる
- 事実の成立度合いをファジィ値で表現
- しきい値より大きな事実が選ばれる
  - モードにより
    - しきい値より大きいもの、すべてが選ばれたり
    - しきい値を越える中でもっとも大きいものが選ばれたり
    - 参考: http://ja.wikipedia.org/wiki/ファジィ論理
      - http://ci.nii.ac.jp/naid/110002724154

- SAN値prologは
  - しきい値より *大きい* ではなく
  - SAN値に *近いもの* を選ぶ

SAN値Prolog プログラム例:(」・ω・)」うー!(／・ω・)／にゃー!

SAN値Prolog

```
90:foo("(」・ω・)」うー").
60:foo("(／・ω・)／にゃー!").
30:foo("xxx").
```

SAN値Prolog , S式版

```
((90 foo "(」・ω・)」うー"))
((60 foo "(／・ω・)／にゃー!"))
((30 foo "xxx" ))
```

SAN値が90ぐらいのときに選ばれる

SAN値が60ぐらいのときに選ばれる

SAN値が30ぐらいのときに選ばれる

SAN度

SAN値Prolog 実行例 1-1:(」・ω・)」うー!(／・ω・)／にゃー!

```
にゃ?-
にゃ?-(set-sanchi 90)
SAN値=20
lisp value=90
にゃ?-((foo *x))
SAN値=90
 - new-head=(FOO *X) sanchi=0
 - choiced-clause=((0 FOO *X))
 - new-head=(FOO "(」・ω・)」うー") sanchi=90
 - choiced-clause=((90 FOO "(」・ω・)」うー"))
 - new-head=(FOO "(／・ω・)／にゃー!") sanchi=60
 - new-head=(FOO "xxx") sanchi=30

result=T
reduced-goal=((FOO "(」・ω・)」うー"))

にゃ?-
```

プロンプト

SAN値を90に設定

直前のSAN値(気にしない)

問い合わせの実行

SAN度90の節を選んで終了

SAN値Prolog実行中

“(」・ω・)」うー” の節が、最終的に選ばれた

SAN値Prolog 実行例 1-2:(」・ω・)」うー!(／・ω・)／にゃー!

```
にゃ?-
にゃ?-(set-sanchi 60)
SAN値=90
lisp value=60
にゃ?-((foo *x))
SAN値=60
 - new-head=(FOO *X) sanchi=0
 - choiced-clause=((0 FOO *X))
 - new-head=(FOO "(」・ω・)」うー") sanchi=90
 - choiced-clause=((90 FOO "(」・ω・)」うー"))
 - new-head=(FOO "(／・ω・)／にゃー!") sanchi=60
 - choiced-clause=((60 FOO "(／・ω・)／にゃー!"))
 - new-head=(FOO "xxx") sanchi=30

result=T
reduced-goal=((FOO "(／・ω・)／にゃー!"))

にゃ?-
```

プロンプト

SAN値を60に設定

直前のSAN値(気にしない)

問い合わせの実行

まず、SAN度90の節を選んだ

SAN値Prolog実行中

SAN度60の節の方が
SAN値に近いので、
それが選ばれた

"(／・ω・)／にゃー!" の節が、最終的に選ばれた

SAN値Prolog プログラム例:足し算

SAN値Prolog

```
90:plus(*X, *Y, *Z) :- *Z = *X + *Y .
60:plus(*X, *Y, *Z) :- *Z1 = *X + *Y,  *Z = *Z1 +1 .
30:plus(1,1, 200)  :- *Z = *X  ×  *Y .
```

SAN値Prolog , S式版

```
((90 plus *x *y *z)(+ *x *y *z))
((60 plus *x *y *z)(+ *x *y *z1)(+ *z1 1 *z))
((30 plus *x *y *z)(mul *x *y *z))
```

SAN値が90ぐらいのときに選ばれる
正常な足し算

SAN値が60ぐらいのときに
選ばれる
普通より1多い結果を返す

SAN値が30ぐらいのときに
選ばれる
掛け算をやっている

SAN度

SAN値Prolog 実行例 2-1:足し算 SAN値=90

```
にゃ?-(set-sanchi 90)
SAN値=61
lisp value=90
にゃ?-((plus 2 6 *x))
SAN値=90
 - new-head=(+ *X *Y *Z) sanchi=1000
 - choiced-clause=((1000 + 2 6 8) (*SYSTEM-FUNC* PRO+ 2 6 8))
 - new-head=(PLUS *X *Y *Z) sanchi=90
 - choiced-clause=((90 PLUS 2 6 8) (+ 2 6 8))
 - new-head=(+ *X *Y *Z) sanchi=1000
 - choiced-clause=((1000 + 8 1 9) (*SYSTEM-FUNC* PRO+ 8 1 9))
 - new-head=(+ *X *Y *Z) sanchi=1000
 - choiced-clause=((1000 + 2 6 8) (*SYSTEM-FUNC* PRO+ 2 6 8))
 - new-head=(PLUS *X *Y *Z) sanchi=60
 - new-head=(MUL *X *Y *Z) sanchi=1000
 - choiced-clause=((1000 MUL 2 6 12) (*SYSTEM-FUNC* PRO* 2 6 12))
 - new-head=(PLUS *X *Y *Z) sanchi=30

result=T
reduced-goal=((PLUS 2 6 8))
```

SAN値を90に設定

2+6=? 問い合わせの実行

SAN度90の節を選んで終了

サブルーチンも同じルールで実行されている
気にしなくてよい

SAN値Prolog実行中

SAN度60の節も評価しているが、
SAN値から遠いので、選ばれない

2+6=8 が得られた

SAN値Prolog 実行例 2-2:足し算　SAN値=65

```
にゃ?-(set-sanchi 65)
SAN値=90
lisp value=65
にゃ?-((plus 2 6 *x))
SAN値=65
 - new-head=(+ *X *Y *Z) sanchi=1000
 - choiced-clause=((1000 + 2 6 8) (*SYSTEM-FUNC* PRO+ 2 6 8))
 - new-head=(PLUS *X *Y *Z) sanchi=90
 - choiced-clause=((90 PLUS 2 6 8) (+ 2 6 8))
 - new-head=(+ *X *Y *Z) sanchi=1000
 - choiced-clause=((1000 + 8 1 9) (*SYSTEM-FUNC* PRO+ 8 1 9))
 - new-head=(+ *X *Y *Z) sanchi=1000
 - choiced-clause=((1000 + 2 6 8) (*SYSTEM-FUNC* PRO+ 2 6 8))
 - new-head=(PLUS *X *Y *Z) sanchi=60
 - choiced-clause=((60 PLUS 2 6 9) (+ 2 6 8) (+ 8 1 9))
 - new-head=(MUL *X *Y *Z) sanchi=1000
 - choiced-clause=((1000 MUL 2 6 12) (*SYSTEM-FUNC* PRO* 2 6 12))
 - new-head=(PLUS *X *Y *Z) sanchi=30

result=T
reduced-goal=((PLUS 2 6 9))
```

SAN値を65に設定

2+6=? 問い合わせの実行

まず、SAN度90の節を選んだ

サブルーチンも同じルールで実行されている
気にしなくてよい

SAN値Prolog実行中

SAN度60の節を選んだ

2+6=9 が得られた。(^^;

SAN値Prolog 実行例 2-3:足し算　SAN値=32

```
にゃ?-(set-sanchi 32)
SAN値=32
lisp value=32
にゃ?-((plus 2 6 *x))
SAN値=32
 - new-head=(+ *X *Y *Z) sanchi=1000
 - choiced-clause=((1000 + 2 6 8) (*SYSTEM-FUNC* PRO+ 2 6 8))
 - new-head=(PLUS *X *Y *Z) sanchi=90
 - choiced-clause=((90 PLUS 2 6 8) (+ 2 6 8))
 - new-head=(+ *X *Y *Z) sanchi=1000
 - choiced-clause=((1000 + 8 1 9) (*SYSTEM-FUNC* PRO+ 8 1 9))
 - new-head=(+ *X *Y *Z) sanchi=1000
 - choiced-clause=((1000 + 2 6 8) (*SYSTEM-FUNC* PRO+ 2 6 8))
 - new-head=(PLUS *X *Y *Z) sanchi=60
 - choiced-clause=((60 PLUS 2 6 9) (+ 2 6 8) (+ 8 1 9))
 - new-head=(MUL *X *Y *Z) sanchi=1000
 - choiced-clause=((1000 MUL 2 6 12) (*SYSTEM-FUNC* PRO* 2 6 12))
 - new-head=(PLUS *X *Y *Z) sanchi=30
 - choiced-clause=((30 PLUS 2 6 12) (MUL 2 6 12))

result=T
reduced-goal=((PLUS 2 6 12))
```

SAN値を32に設定

2+6=? 問い合わせの実行

まず、SAN度90の節を選んだ

SAN値Prolog実行中

SAN度60の節を選んだ

SAN度30の節を選んだ

最終的に 2+6=12 が得られた。掛け算やってる(^^;

おしまい

- http://www.takeoka.org/~take/
- www.takelab.com
- www.axe.bz