

きふわらべ アピール文書 (大会後差替)

2023年05月07日 高橋智史

SANSYO

3勝 したぜ

JIKO

実質的に、自己ハイ

順位	名前	1×	21×	15×	5×	21×	17×	9×	25×	0	0	0	20.0	0.0	0.0	20
23	きふわらべ	6×	20×	180	8×	16×	270	19×	220	3	5	0	21.0	4.0	1.0	25
24	HoneyWaffle	50	190	30	12×	10	20	28×	70	6	2	0	41.0	27.0	18.0	4



開発者
高橋 智史 (著作権上、問題ありません)

「 **きふわるべ**(※1) と、**きふわらべ軍団**(※2) を除くと自己ハイの3勝だぜ。 3勝5敗」

※1 … 2016年5月 第26回世界コンピュータ将棋選手権(WCSC26)に参加した、きふわらべの通称。
わたしの中での識別名称は きふわらぷりー。
4勝3敗。「大樹の枝」ライブラリーを使用。

※2 … 2020年10月 第1回世界将棋AI電竜戦に参加した、複数のきふわらべの通称。
わたしの中での呼称は きふわらべーず。 9体で参加して兄弟を倒して星を増やした。 一番強いきふわらべは 3勝7敗。



コンピュータ将棋エンジン
きふわらべ (著作権上、問題ありません)

「3回戦の **重力場計算法** は 1手詰めを見逃してくれて、
6回戦の **JHBR** は こっちが王手したら 投了してくれたぜ」

フロム・スクラッチ宣言

思考部に大きな影響を与える、他者の作成したプログラム・データ等を利用していません。

フリーフォント「ためき油性マジック」 作者：ためき侍 (利用ライセンス上、問題ありません)

<https://tanukifont.com/tanuki-permanent-marker/>



ひよ子 (著作権上、問題ありません)

「 8回戦の 臥龍 には 頭金を決めてるわよ」



「 きふわらべが 将棋のルールで まともに勝ったのは WCSC 31の ponkotsu 戦に続いて 2例目だけ」



「 わたしは 持ち駒を打って 利きを当てれば 駒を取れると思ってるから、それが たまたま王手で、 飛車も利いていたのが ラッキーだったぜ」

わたしは 並列 処理を初めて使ってみた

SHI-PI-YU-

並列処理は、複数の CPU を同時に使うこと

第29回世界コンピュータ将棋選手権 きふわらべ アピール文書

<https://www.apply.computer-shogi.org/wcsc29/appeal/Kifuwarabe/appeal.pdf>


「👉 4年前の WCSC29 出場時から
Ryzen Threadripper 2920X
12コア を持ってるんだが、

パラレル・プロセッシング
(Parallel processing ; 並列処理)
のプログラムの書き方を 知らなかった」



「新しいものは 買っただけで
満足するタイプですからね」



「👉 エリクサー (Elixir) 言語が
パラレル・プロセッシングを
得意としてて、
Flow ライブラリーを使うだけで
いいぞうだぜ」



```

28 move_list_on_board =
29   if remain == 1 or remain == 2 do
30     #
31     # 盤上の自駒
32     # -----
33     #
34     # ▷ スペース (:sp; 空マス) は除去。かつ、手番の駒だけ残す
35     # ▷ (並列化) `Flow.from_enumerable()` - コンピューターの各コアへ処理を振り分け
36     # ▷ (並列化) ピース (Piece; 先後付きの駒種類) から、先後を消し、ピース・タイプ (Piece Type; 駒種類) に変換する
37     #     マス番地と駒種類から、指し手生成
38     # ▷ (並列化終り) `Enum.to_list()` - 振り分けた処理をリストに戻す
39     # ▷ リストがネストしていたら、フラットにする
40     # ▷ 指し手が nil なら除去
41     move_list_on_board =
42       pos.board
43       ▷ Enum.filter(fn {_sq, piece} →
44         piece ≠ :sp and pos.turn == KifuwarabeWcsc33.CLI.Mappings.ToTurn.from_piece(piece)
45       end)
46       ▷ Flow.from_enumerable()
47       ▷ Flow.map(fn {sq, piece} →
48         piece_type = KifuwarabeWcsc33.CLI.Mappings.ToPieceType.from_piece(piece)
49         pos ▷ make_move_list_by_piece_on_board(sq, piece_type)
50       end)
51       ▷ Enum.to_list()
52       ▷ List.flatten()
53       ▷ Enum.filter(fn move → !is_nil(move) end)
54
55     move_list_on_board
56   else

```



「初めて パラレル・プロセッシング を使ってみて どうだったんだけ？」



「低速化した。
2手読み 0秒 だったところ、1分以上かかるようになった。

嫌になって、パラレル・プロセッシングする回数を 減らすことにしたぜ」

- ◆ 今何手目かを10で割って 余りが 1、2 のときは 盤上の駒を動かす指し手 を並列処理で生成する
- ◆ 今何手目かを10で割って 余りが 3、4 のときは 持駒を打つ手 を並列処理で生成する

無題 - 将棋所

ファイル(F) 編集(E) 局面編集(B) 対局(S) 表示(V) ヘルプ(H)

切 | 投 | 待 | 急 | 入 | 中 | 検 | 解 | 詰 |

対局中
先手番

残り時間 加算時間
▲ 人間 00:15:00 5秒
△ Kifuwaraxir 00:15:09 5秒

==== 開始局面 ==== (1手 / 合計)
1 ▲ 2六歩 (27) 00:01 / 00:00:01
2 △ 9四歩 (93) 00:01 / 00:00:01

コメント

更新 消去

4900
4000
3000
2500
2000
1500
1000
500
0
-500
-1000
-1500
-2000
-2500
-3000
-3500
-4000
-4500

△ Kifuwaraxir 予想手: 現在の探索手: 探索深さ: 2 探索局面数: 960 NPS: 7680 ハッシュ使用率:

思考時間	探索深さ	探索局面数	評価値	読み筋
00:00	2	960	0	The remainder of the 2(th) move divided by 10 is 2. Hello Multi-core processor(^_^)! I use Flow module on Elixir. Parallel processing of move generation on the board!



「👉 きふわらべちゃん、なんか 読み筋の欄を使って 何か言ってるわねえ」



「盤上の駒を動かす指し手を **パラレル・プロセッシング**して生成したと言ってるぜ。

WCSCの大会サーバーは **読み筋以外のコメントは受け付けない**ので公式の棋譜には残っていないぜ」



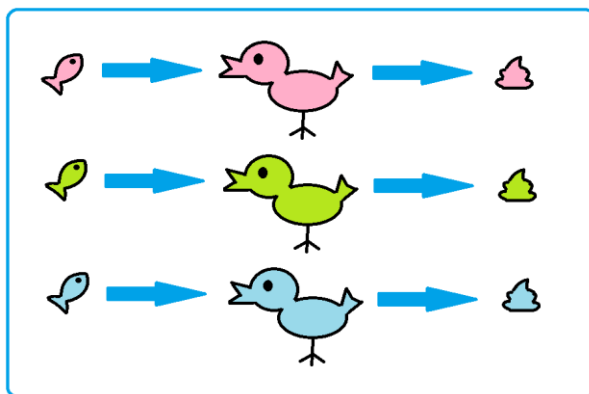
「今まで使ってなかった **スレドリッパー**のマルチコアを使って **パラレル・プロセッシング**できて **満足**だぜ」



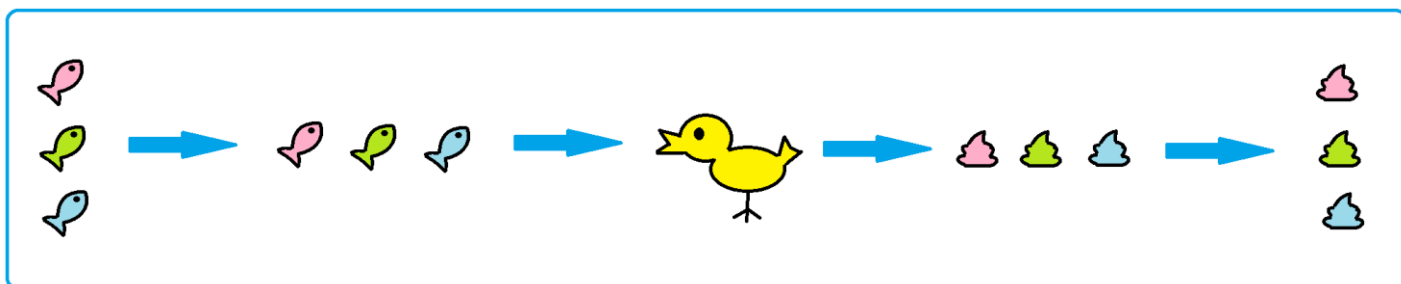
「お父さんが WCSC31、32 で使っていた **ゴ (Go) 言語** には **Flow**モジュールなんかインストールしなくても **最初から **コンカレント・プロセッシング (Concurrent processing ; 並行処理)** の **ゴルーチン (Goroutine)** が付いてる**そうだぜ」



「**パラレル・プロセッシング (Parallel processing ; 並列処理)**と、**コンカレント・プロセッシング (Concurrent processing ; 並行処理)**は **違う**んで」



Parallel processing



Concurrent processing

GI-PI-YU-

わたしは GPU を初めて計算に使った気分になった

GI-PI-GI-PI-YU-

GPGPU は、GPU を計算に使うこと



「新しい GPU を買ったぜ」



「 GeForce RTX 4090 ね」



「 GeForce RTX 3080 Ti を既に持ってたのに なんでもた上位モデルを買う？ダブってるじゃないかだぜ？」



「わたしは **ハッホー** と思えるものを **気軽に** 買う。人生の良い選択とか 決意とか 経済合理性のようなものは **無い**」



「 そうですね」



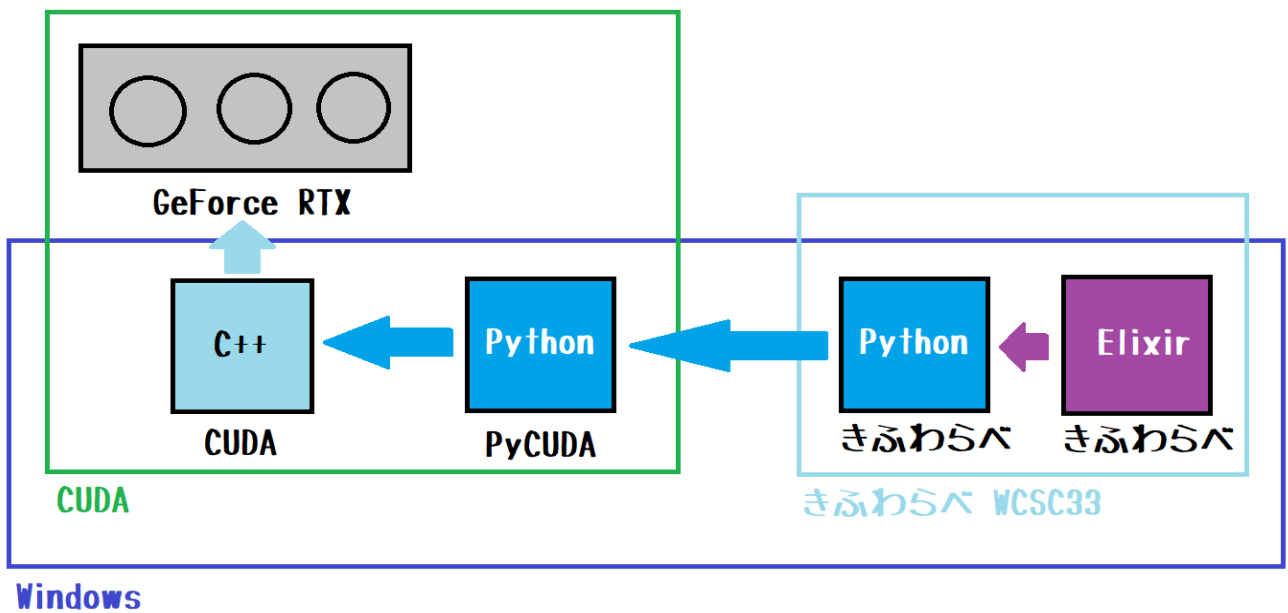
「わたしは お父さんが **ハッホー** したいから作られた AI だしな」



「パイトーチ (PyTorch) で 機械学習のサンプルを動かせば GPUの使い方を知らなくても GPUは使われるわけだが……」



「下がっていく数値や、折れ線グラフを見ても GPUを使っている気分がしない、というのが わたしに 合わなかった。そこで今回、その気分を 上げることにした」



「👉 エヌビディア (NVIDIA) という会社が作っている ジーフォース (GeForce) というイケてるグラフィックボードを使うには、クーダ (CUDA) というライブラリを プログラムから使えばいいんだが」



「クーダは シープラプラ (C++) 言語で書かれていて、きふわらべは エリクサー (Elixir) 言語で書かれていて 言語が違って 直接使うことは できないのだった」



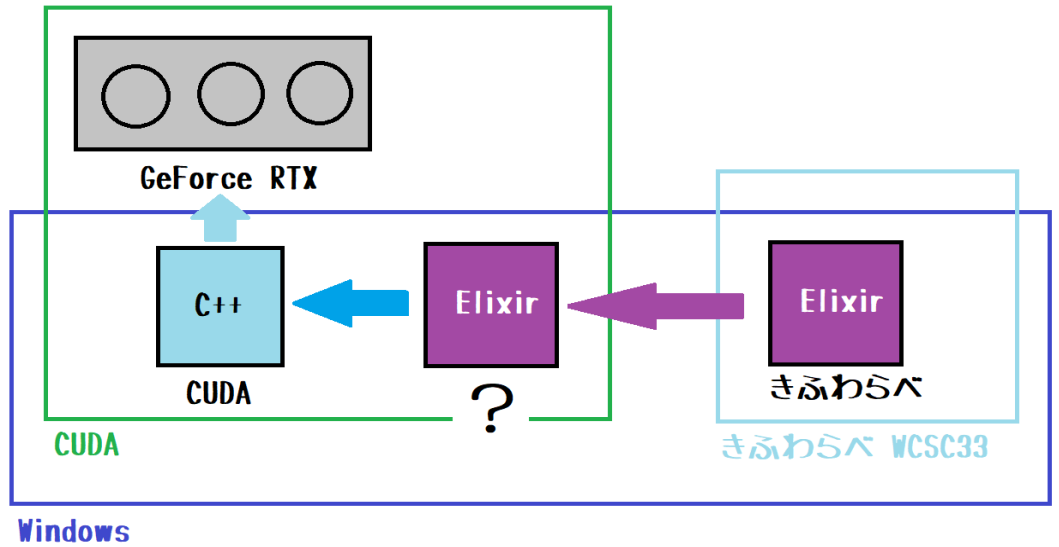
「クーダを直接使ったら 気分が上がるのね」



「シーブプラ はなんか 難しかったので、
パイソン (Python) 言語で書かれているパイ・クーダ (PyCUDA) ライブラリ
に間に入れてもらうことにしたんだぜ」



「じゃあ エリクサーの平行・プロセッシングは
GPGPUには 利用されてなくて、
パイソン 使ってるのと同じなんだ」



「👉 それでは **ハッホー** できないだろ。
エリクサー言語用のクーダ・ライブラリーは無いのかだぜ？」



「Windowsでコンパイルできるものが
見つからなかったぜ。
コンパイル・エラー」



「Linuxディストリビューションに
乗り換えないの？」



「Linuxディストリビューション用に パソコンを もう1台
用意するのも視野に入れ、新しいモデルを買って 古いのをダブらせて
スレッドリッパーと ジーフォースを 1セット 余らせようぜ？」

一度に パソコンの部品を 買い揃えるのは 大変だしな」


```

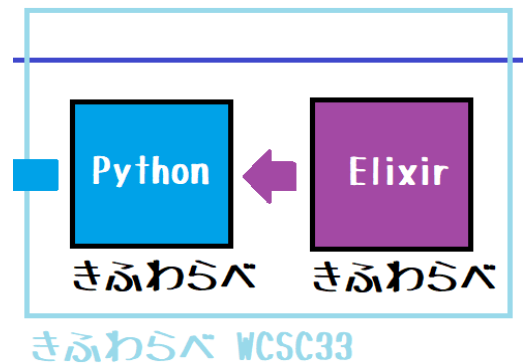
1 defmodule KifuwarabeWcsc33.CLI.CallPython.HelloGpu do
2   def hello_gpu do
3     {_output, _status} = System.cmd("python", ["hello_gpu.py"],
4     [
5       cd: "../lib_python",
6       into: IO.stream(:stdio, :line)
7     ])
8     # IO.puts("finished. status:#{status}")
9     # IO.inspect(output, label: "[hello_gpu] output")
10    # finished. status:0
11    # [hello_gpu] output: %IO.Stream{device: :standard_io, raw: false, line_or_bytes: :line}
12
13    {:ok, self()}
14  end
15 end
16 end
17
18
19

```



「上記が
エリクサーからパイソンを
呼び出している箇所だけ。」

hello_gpu.py というスクリプトを
実行している」



pycuda/hello_gpu.py at main · inducer/pycuda

Search or jump to... Pulls Issues Codespaces Marketplace Explore

Branch master was renamed to main.

inducer / pycuda Public Watch 56 Fork 270 Star 1.5k

Code Issues 65 Pull requests 14 Discussions Actions Projects Security Insights

Beta Try the new code view

main pycuda / examples / hello_gpu.py / <> Jump to Go to file

inducer Run pyupgrade 3.6+ Latest commit f95bf36 on Oct 12, 2020 History

2 contributors

26 lines (21 sloc) | 583 Bytes Raw Blame

```
1 import pycuda.driver as drv
2 import pycuda.tools
3 import pycuda.autoinit
4 import numpy
5 import numpy.linalg as la
6 from pycuda.compiler import SourceModule
7
8 mod = SourceModule("""
9 __global__ void multiply_them(float *dest, float *a, float *b)
10 {
11     const int i = threadIdx.x;
12     dest[i] = a[i] * b[i];
13 }
14 """)
15
16 multiply_them = mod.get_function("multiply_them")
17
18 a = numpy.random.randn(400).astype(numpy.float32)
19 b = numpy.random.randn(400).astype(numpy.float32)
20
21 dest = numpy.zeros_like(a)
22 multiply_them(
23     drv.Out(dest), drv.In(a), drv.In(b),
24     block=(400,1,1))
25
26 print(dest-a*b)
```

Give feedback

Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About

© 2023 GitHub, Inc.

[inducer/pycuda pycuda/examples/hello_gpu.py](https://github.com/inducer/pycuda/blob/main/examples/hello_gpu.py)

https://github.com/inducer/pycuda/blob/main/examples/hello_gpu.py



「満足だぜ」



「大会中に ハロー・ジーピーユー を実行してたの？」



「何手目かの数を10で割った余りが 5か 6のときに
サーチ部に入る前に 1回 実行してるぜ」



「GPUを使ったな」



「使ったのかなあ？」

MITAINA

Elixir は 関数型「みたいな」言語



「今回のきふわらべは エリクサー (Elixir) 言語で実装したぜ。
ハロー・ジーピーユーを呼び出すとこだけ パイソン (Python) 言語を
使ったけど」



「エリクサーには どんな利点があるの？」



「利点なんか どうでもいいんだぜ。
関数型言語を使ったから 満足だぜ」



「ガチの関数型言語は ハスケル (Haskell) が有名なんだが
Windows では環境設定すら難しかったので ハスケルは止めた」

イベント	カテゴリ	CPU / メモリ	クロック	1	+	OS	OS	CTT	リンク
ながとダイアリー	Haskell将棋	Athlon 2650e	1.6GHz	1	1	1GB	WindowsXP	Haskell	Bonanza リンク

第21回世界コンピュータ将棋選手権

<http://www2.computer-shogi.org/wcsc21/>


「 WCSC21 に Haskell将棋 (ながとダイアリー) というのが
ハスケル言語で参加しているのを 大会帰りの電車で教えてもらったんで
きふわらべは 関数型言語で最初に参加したソフトではないぜ」



「 CSAのWebサイトをぱっと見た感じ、
WCSCの参加者で
エリクサーで実装された将棋エンジンは わたしが初じゃないかな」


```

1  defmodule KifuwarabeWcsc33.CLI.Search.Alpha do
2    # 探索 (深さへ再帰)
3    def do_it(pos, depth, nodes_num_searched \\ 0) do
4      move_list = KifuwarabeWcsc33.CLI.MoveGeneration.MakeList.do_it(pos)
5      {move_list, pos} = KifuwarabeWcsc33.CLI.MoveList.ReduceSuicideMove.do_it(move_list, pos, :teban)
6      {pos, best_move, value, nodes_num_searched} = choice_best(pos, move_list, nil, -32768, depth, nodes_num_searched)
7      {pos, best_move, value, nodes_num_searched}
8    end
9
10   # 探索 (水平へ再帰)
11   defp choice_best(pos, move_list, sibling_best_move, sibling_best_value, depth, nodes_num_searched)
12
13   defp choice_best(pos, [], sibling_best_move, sibling_best_value, _depth, nodes_num_searched) do
14     {pos, sibling_best_move, sibling_best_value, nodes_num_searched}
15   end
16
17   defp choice_best(pos, [move | move_list], sibling_best_move, sibling_best_value, depth, nodes_num_searched) do
18     pos = pos > KifuwarabeWcsc33.CLI.MoveOperation.DoMove.do_it(move)
19     nodes_num_searched = nodes_num_searched + 1
20
21     {pos, opponent_value, nodes_num_searched} =
22       if depth < 2 do
23         opponent_value = lets_position_value(pos)
24         {pos, opponent_value, nodes_num_searched}
25       else
26         {pos, _best_move, opponent_value, nodes_num_searched} = do_it(pos, depth - 1, nodes_num_searched)
27         {pos, opponent_value, nodes_num_searched}
28       end
29
30     value = -opponent_value
31
32     {sibling_best_move, sibling_best_value} =
33       if is_alpha_update?(value, sibling_best_value) do
34         is_uchifu_dume? = KifuwarabeWcsc33.CLI.Thesis.IsUchifuDume.is_uchifu_dume?(move, pos)
35
36         if is_uchifu_dume? do
37           {sibling_best_move, sibling_best_value}
38         else
39           {move, value}
40         end
41       else
42         {sibling_best_move, sibling_best_value}
43       end
44
45     # 再帰 (深さへ再帰)
46     pos = pos > KifuwarabeWcsc33.CLI.MoveOperation.UndoMove.do_it()
47
48     if move_list > length() < 1 do
49       {pos, sibling_best_move, sibling_best_value, nodes_num_searched}
50     else
51       # 再帰 (水平へ再帰)
52       choice_best(pos, move_list, sibling_best_move, sibling_best_value, depth, nodes_num_searched)
53     end
54   end
55
56   defp is_alpha_update?(value, sibling_best_value) do
57     sibling_best_value < value
58   end
59
60   defp lets_position_value(pos) do
61     pos.materials_value
62   end
63 end

```



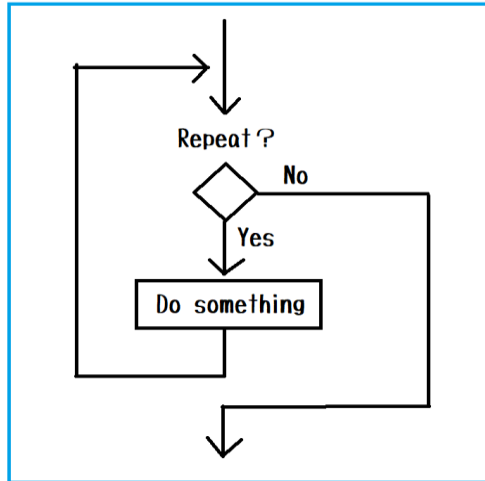
「👉 エリクサーで実装した サーチ部だけ。

インターネット上に エリクサーで実装したコンピューター将棋の
 エグザンプルが無いから 苦しんだ」



「エリクサーって
どんな言語なの？」

```
// C
for ( int i = 0; i < 10; i++ )
{
    printf( "i=%d\n", i );
}
```



```
// C
int i = 0;
while( i < 10 )
{
    printf( "i=%d\n", i );
    i++;
}
```

```
# Elixir
def do_repeat(i)

def do_repeat(i) when 10 <= i do
  nil
end
```

Loop

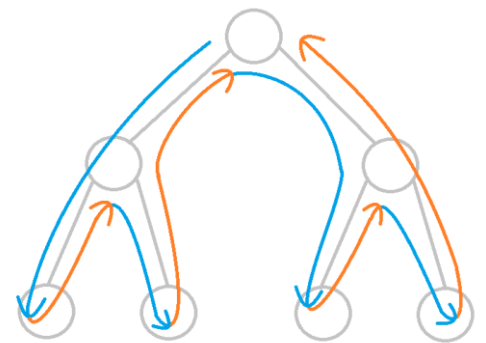


「☞ フォー（for）ループも
ホワイル（while）ループもない。
ループは再帰関数で
書かないといけない」

```
def do_repeat(i < 0) when i < 10 do
  IO.puts("i=#{i}")
  do_repeat(i + 1)
end
```

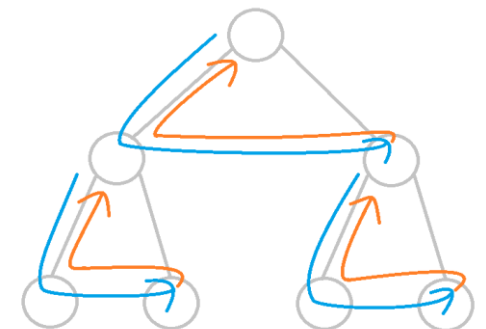


「ぜったい 関数でやってやる
という やる気を感じるぜ」



「☞ 深さの方へ
再帰するのは
よくやるが……」

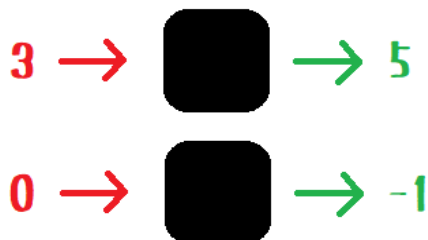
青い線：行き
オレンジ色の線：帰り



「☞ 水平にも再帰でやらな
ければいけないの
初めてだぜ」



「なんで むりやり なんでもかんでも 関数でやるの？」



$$x \times 2 - 1$$

Function



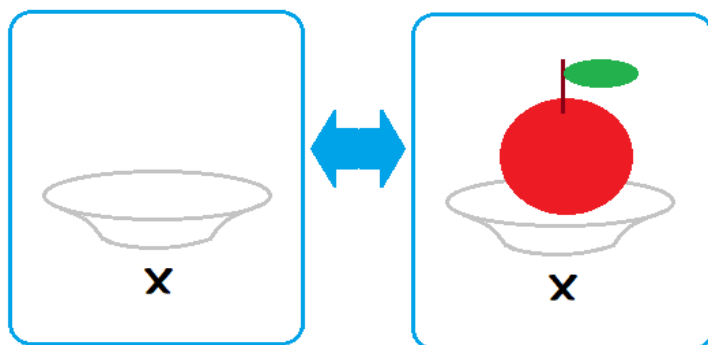
「☞ 入力に対応する出力は 常に同じ、というのが関数型言語の 特徴だけ」



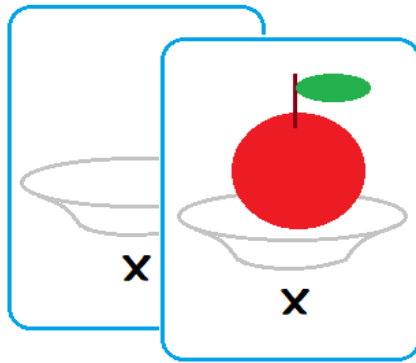
「内部状態を持たない、とも言うぜ」



「ローカル変数もないの？」



「変数というのは Xという皿に 何もない状態、りんごがある状態と、例えば この場合 2つの状態を持つが……」



「エリクサーでは、同じ名前で再定義する、という理屈で
さっきの X と新しい X は名前が同じだけで 別物だから
X は 状態は持ってないんだ と言い張ってるな」



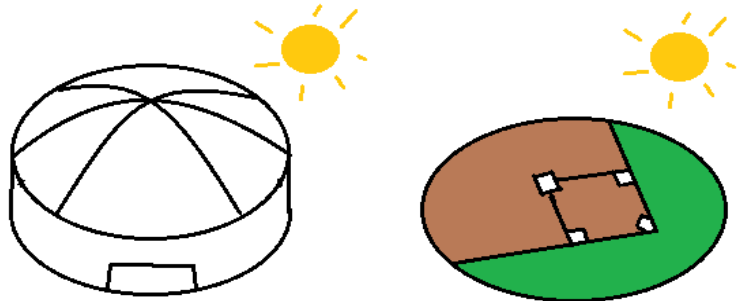
「それはそれで メモリーの使用量を気にするなあ。
内部的に どう メモリーが使用されてるかまで 説明してくれなきゃ
コンピューター将棋のサーチ部の実装は やってられないのよ」



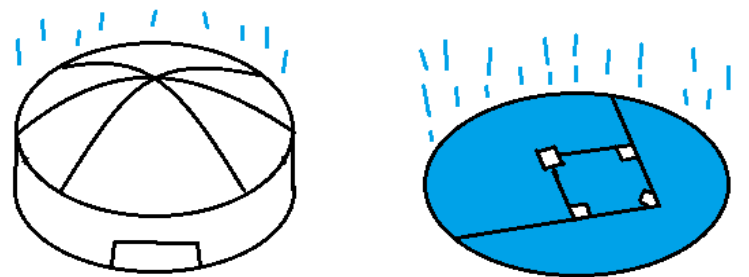
「なんでそんなに 状態を嫌うんだぜ？」



「☞ 天気は状態だぜ。
雨が降ったら
野球は中止だぜ」



「ドームなら
天気という状態は
考えなくて済むだろ」



「考えることを
少なくしたいのか」



「不具合の原因が分からなくて 探すときなんかは
考えることが少ない方がいいわねえ」

プログラミングで ハッホー しようぜ？

SYURUI

6種類の プログラミング・パラダイム



「わたしが いつの頃だったか、
海外の伝説的なプログラマーが、誰だったか忘れたが、言ったことが
記事になったものを読んで 頭に残ってるんだぜ」

プログラマーなら 少なくとも
6種類のプログラミング言語を やるといい

プロシージャ型言語
オブジェクト指向型言語
関数型言語
論理型言語



「あとは何だったか 忘れた」



「覚えてるよ！」



「プログラミング・パラダイム (Programming paradigm) で
検索すると、
6個どころではなく 沢山出てくるんだけど」



「色々あるという ヒント が示されただけで 充分だぜ」



「今大会のわたしは
CPUの 平行・プロセッシング を使うのと、
GPUの 平行・プロセッシング を使うのと、
プログラミング・パラダイムの 関数型言語 を使うという
3つの使うの **ハッポー** が達成できた実りの多い大会だったな」



「達成なのかなあ？」

Git Hub きふわらべWCSC33

<https://github.com/muzudho/kifuwarabe-wcsc33-written-in-elixir-for-windows>



「👉 この Git Hub の 266コミット目が 今大会に出た きふわらべ だぜ。
コードは MITライセンス にしてある。好きに使えだぜ。
わたしの中での識別名は **きふわらくさ (Kifuaraxir)** だぜ」



「草」

```
>>> Dad, I want you to use
      the computer
you bought to the fullest
      (Grass)
```