ICFPプログラミングコンテスト舞台裏

稲葉 一浩 (kinaba)

on behalf of the ICFP-PC '11 Organizers

ACM SIGPLAN International Conference on Functional Programming

ICFP Programming Contest

- ICFP の主催するプログラミングコンテスト
- 6月か7月に開催 (の年が多い)
- 72時間勝負 (の年が多い)
- チーム人数制限なし (の年が多い)

ICFP Programming Contest

- 使用言語は自由（関数型言語に限りません！）

In addition (to 賞金), the organizers will declare …

- *the first place team's language is "the programming language of choice for discriminating hackers",*
- *the second place team's language is "a fine tool for many applications", and …*

ICFP Programming Contest

- コンテストで競うテーマは？
  - 年によってさまざま。開始と同時に公開
  - 関数型と無関係なことの方が多い
  - Organizer が自由に決める

歴代のテーマ

対戦ゲーム

1998 ○×ゲーム
2002 対戦型倉庫番
2004 アリ
2005 スコットランドヤード
2011 Lambda: the Gathering

最適化

1999 ノベルゲーム
2000 レイトレーシング
2001 HTML
2003 カーレース
2008 火星探査
2009 衛星

その他

2006 UMIX
2007 遠藤さん
2010 車と燃料

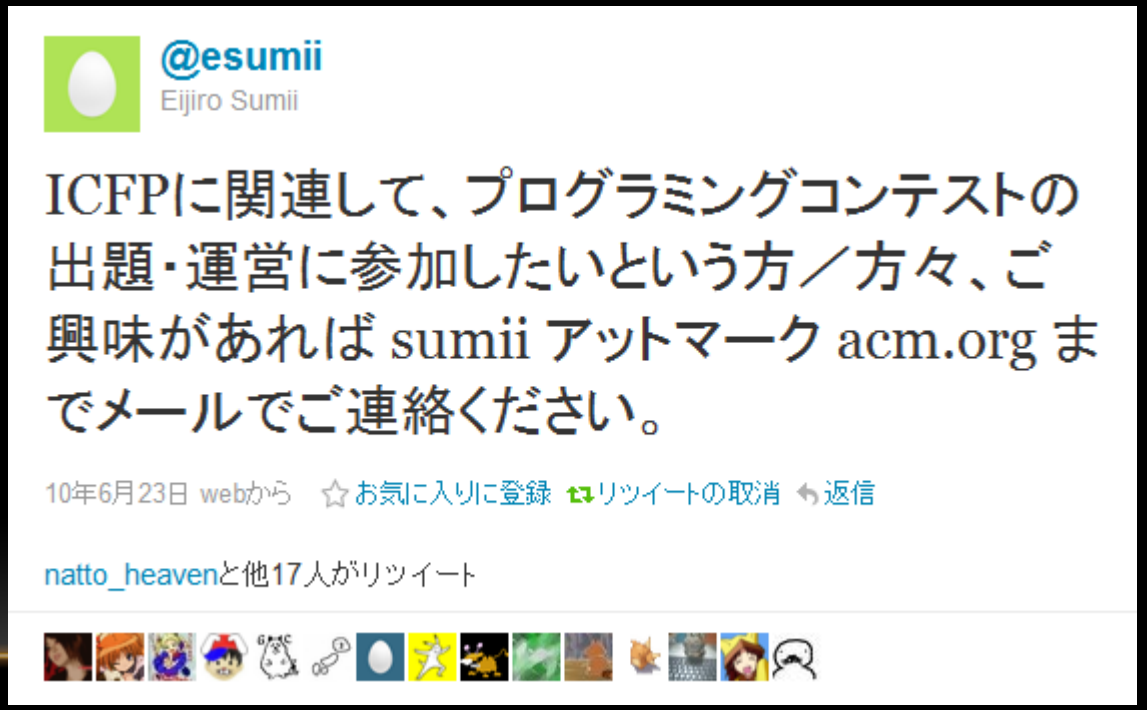
時系列順に振り返る

今年のICFPコンテスト

2010年6月23日（昨年のコンテストの翌々日）

- Contest Chair : E. Sumii

Twitterで
運営チーム
募集



前年6月～11月：運営チーム結成

[Chair] E. Sumii

[Observers] K. Asai, A. Igarashi, Y. Minamide

[Members] H. Abe, Y. Arai, N. Hirota, R. Ina, K. Inaba, A. Iwai, C. Kaneko, S. Kawanaka, M. Masuko, R. Sato, Y. Shibata, Y. Sugawara, T. Tsukada, K. Tsushima, Y. Ueda

➔ 東北大、お茶大、京大、筑波大、東大、Google

前年7月 ～ 2月 : メーリングリストでアイデア投げ

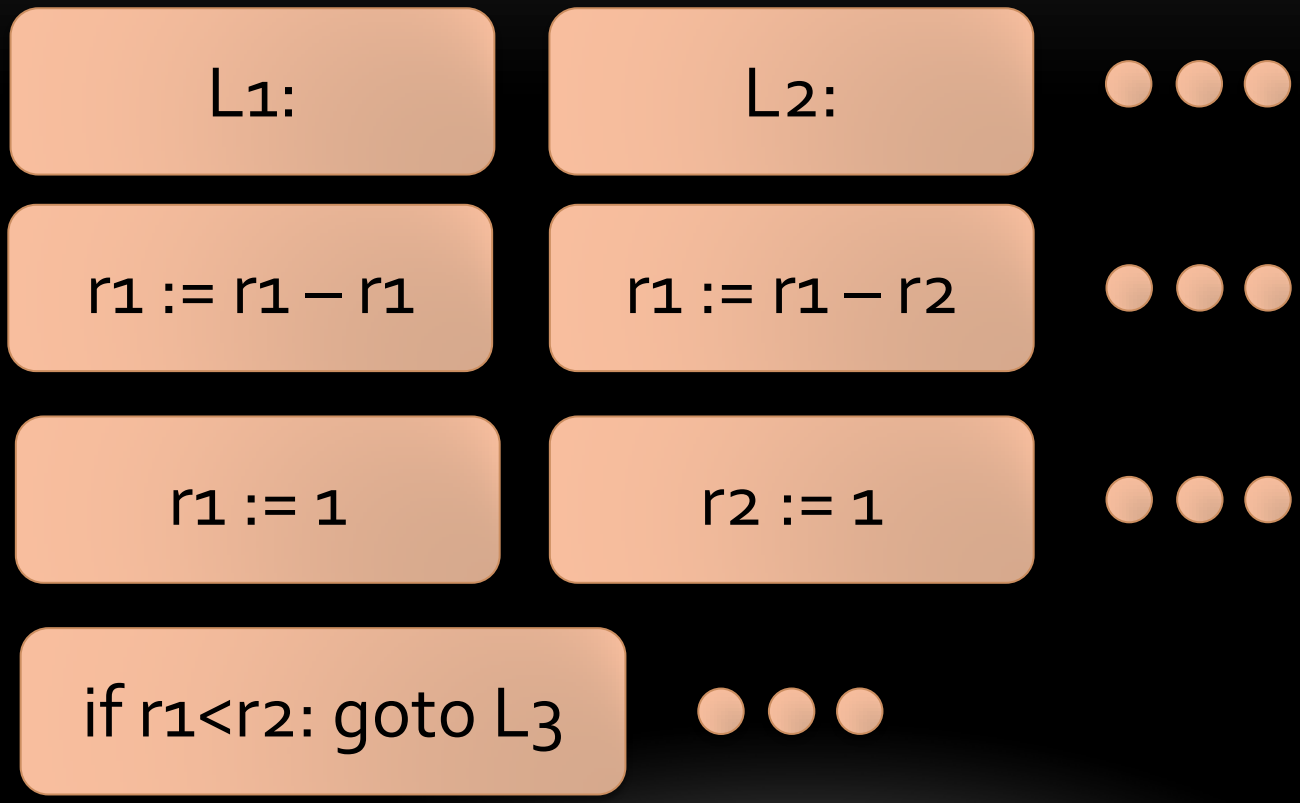
- “Core Wars” の構造化言語バージョン
- 証明体系と自動証明器投げつけ合い
- バグ入り難読言語を最短の変更で直す勝負
- カードにプログラム片が書いてあるカードゲーム
- ・・・

2月26日：ver. 1 “play.ml”

「２人のプレーヤがプログラム断片の書かれたカードを交互に出し合い、完成したプログラムの実行結果（特定の整数変数の値）で勝敗を決める、というもの」

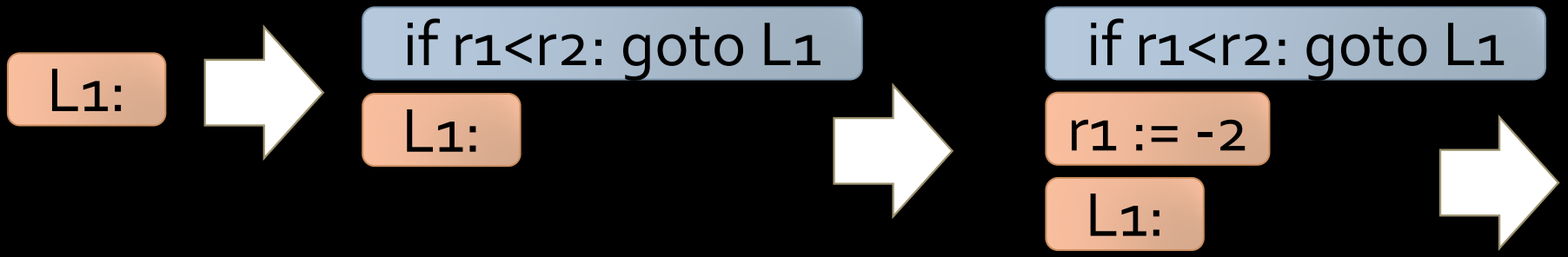
のプレイヤープログラムを書いて競うコンテスト

ver 1: 減算、定数代入、比較ジャンプ命令のみのレジスタマシン



ver 1: 減算、定数代入、比較ジャンプ命令のみのレジスタマシン

- プレイヤは交互にカード列にカードを挿入



- 最終ターンのプログラムを動かして r1 の正負で勝敗判定

3月9日～11日

- 対面ミーティング

第13回プログラミングおよびプログラミング言語ワークショップ

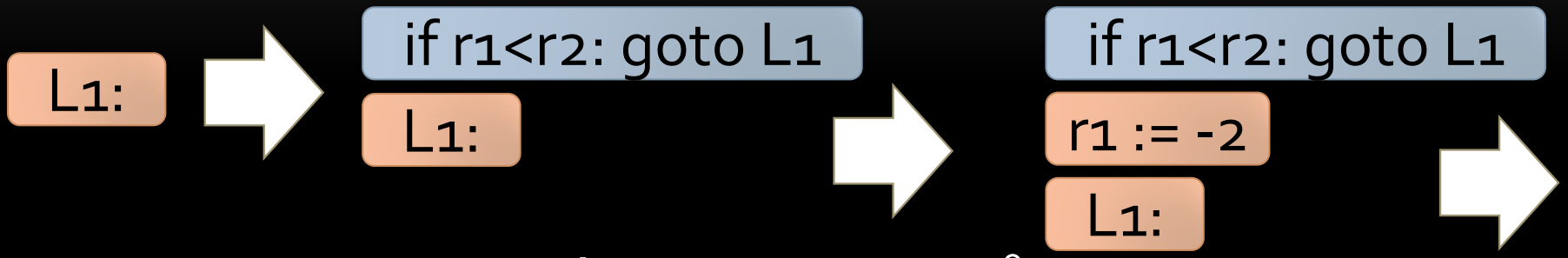
PPL2011

日程: 2011年3月9日(水) ～ 11日(金)
会場: 北海道札幌市 定山渓ビューホテル
主催: 日本ソフトウェア科学会 プログラミング論研究会

ICFPコンテストの問題に要求されること

- 「72時間おもしろさが保つ」こと
- 解き方にバリエーションが出ること
  - × 簡単すぎて“最適解”に収束
  - × 難しすぎて、ランダムな探索を高速化するくらいしかすることがない

議題：このゲームは難しすぎる



- カード10枚程度でも人手でプレイは
  ほぼ不可能
- 「何もヒューリスティックを入れず
  完全ランダムにモンテカルロ木探索」
  以上の解が72時間では見つからなさそう

なにがまずいか？

- 手が広すぎる。2～3手先を読むことすら難しい
- 後手が最後の手でちゃぶ台返しできてしまう。

- 最後に一度だけ"実行"というのが問題
  - 先が読みにくい。
    毎ターンプログラムが実行されるタイプにした方がよい
- 同じメモリ空間を共有するのもまずい
  - 相手にすぐに邪魔されるので、まともなコードが組めない

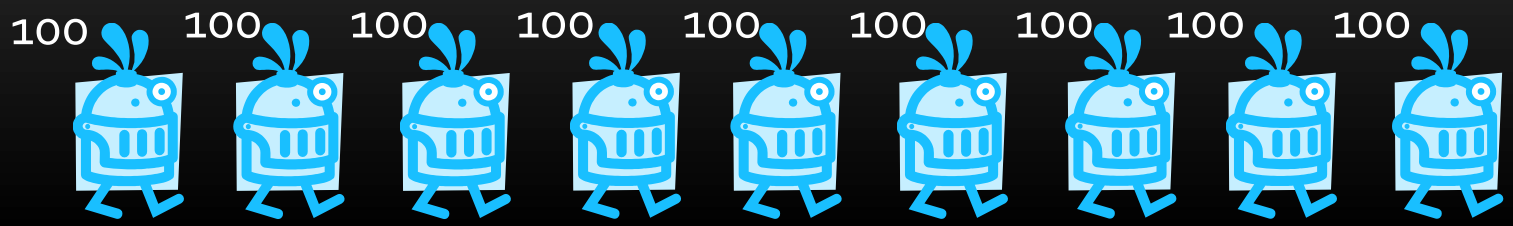
3月23日 : ver 2. “lambda.ml”

3月23日 : ver 2. “lambda.ml”

- カードには関数や自然数が書いてある。
- 場にも関数や自然数が置いてある。

  「カードを出す ＝ 関数適用」
- を毎ターン実行
- 制御構造もラベルジャンプではなく関数で！

256匹のモンスター(HP:100)と、4個の関数スロット



fun x
=> x

fun x
=> x

fun x
=> x

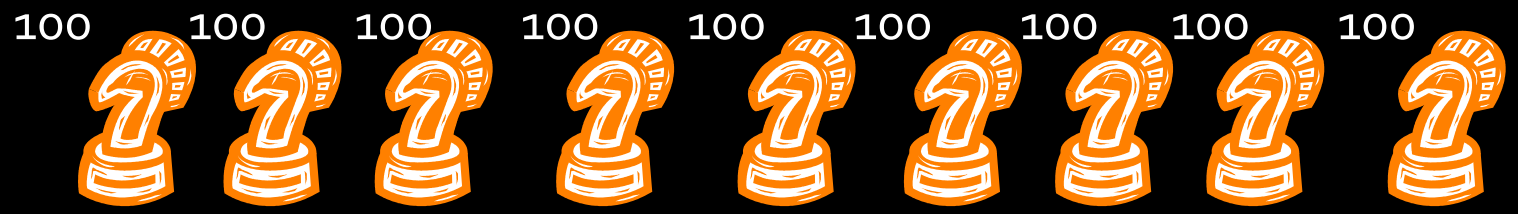
fun x
=> x

fun x
=> x

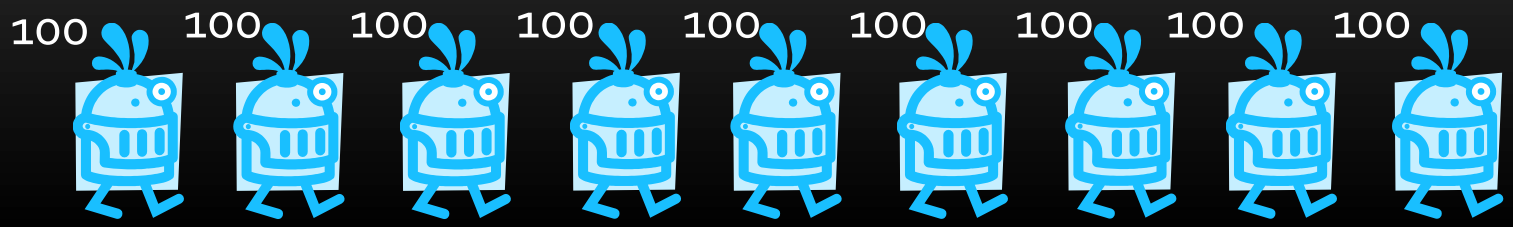
fun x
=> x

fun x
=> x

fun x
=> x



毎ターン１枚、関数カードを場に出す



fun x
=> x

fun x
=> x

fun x
=> x

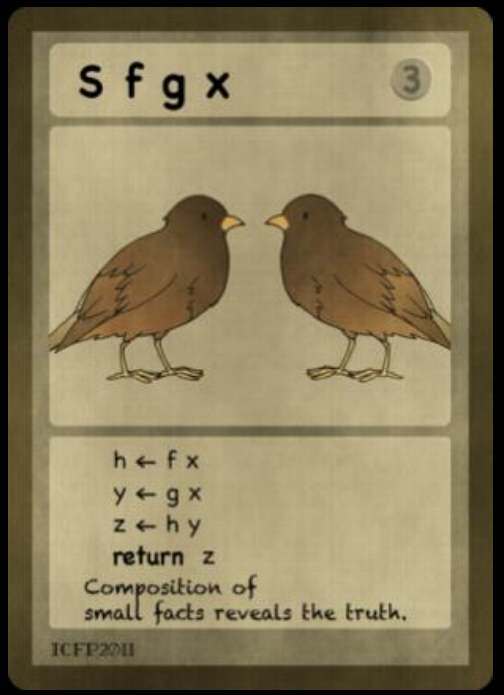
fun x
=> x

出し方は３通り

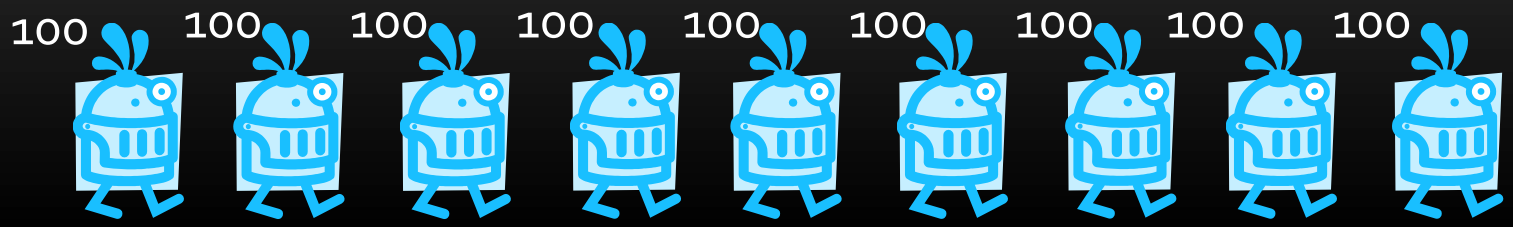
[put]　　スロット := カード

[right]　　スロット := スロット (カード)

[left]　　スロット := カード (スロット)



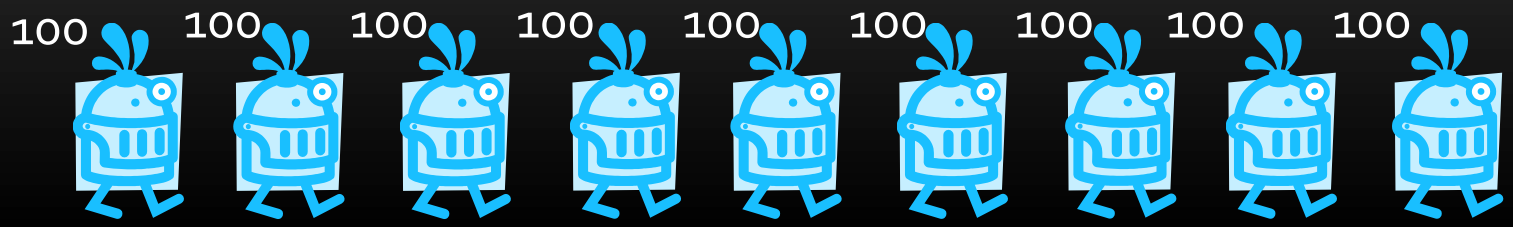
256匹のモンスター(HP:100)と、4個の関数スロット



s1 := s1(zero)

fun x => x	0	fun x => x	fun x => x
fun x => x	fun x => x	fun x => x	fun x => x

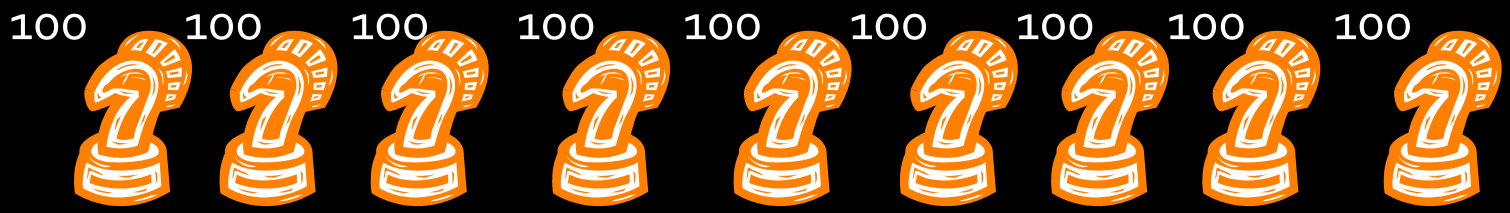
256匹のモンスター(HP:100)と、4個の関数スロット



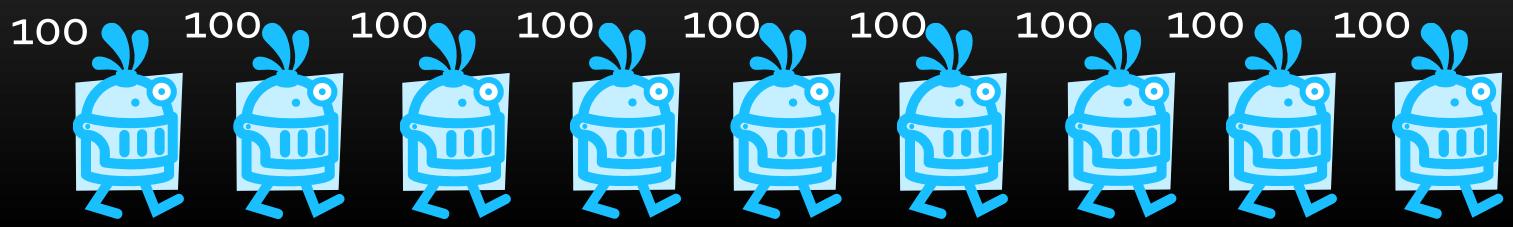
fun x => x	0	fun x => x	fun x => x

S3 := succ

fun x => x	fun x => x	fun x => x	x => x+1

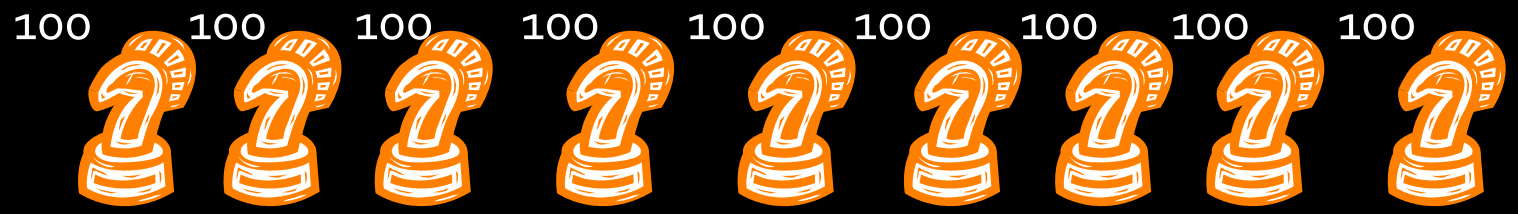


256匹のモンスター(HP:100)と、4個の関数スロット

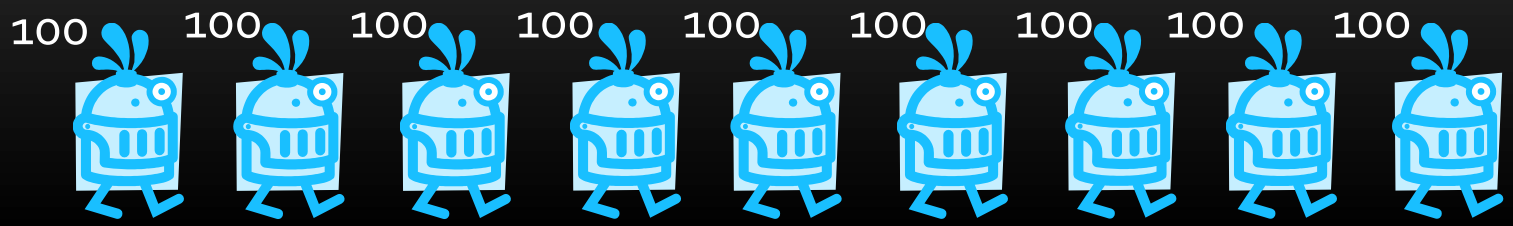


S1 := SUCC(S1)

fun x => x	1	fun x => x	fun x => x
fun x => x	fun x => x	fun x => x	x => x+1



256匹のモンスター(HP:100)と、4個の関数スロット

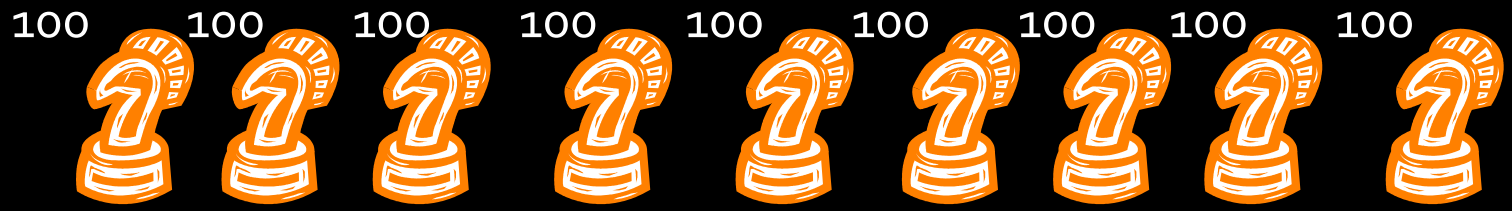


fun x => x	1	fun x => x	fun x => x

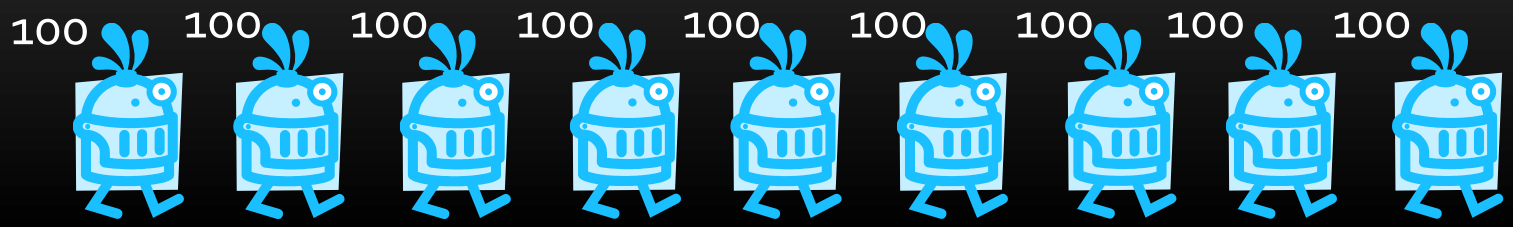
ERROR! id関数に戻る

fun x => x	fun x => x	fun x => x	fun x => x

s3 := zero(s3)



256匹のモンスター(HP:100)と、4個の関数スロット

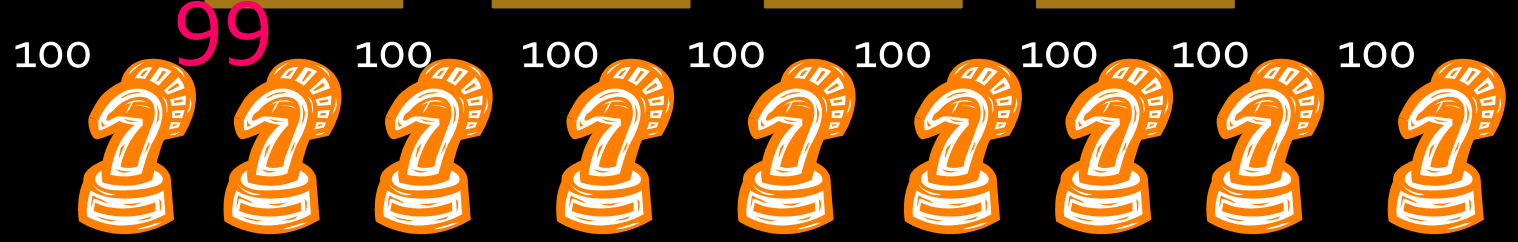


s1 := dec(s1)

fun x => x	fun x => x	fun x => x	fun x => x

副作用カード。相手のs1番のモンスターに1ダメージ！

fun x => x	fun x => x	fun x => x	fun x => x



ICFPコンテストの問題に要求されること（再掲）

- 「72時間おもしろさが保つ」こと
- 解き方にバリエーションが出ること
  - × 簡単すぎて “最適解” に収束
  - × 難しすぎて、どうしようもない

議論：2月～5月

- どのようなカードがあるべきか？
  - プレイヤーに独自カードをデザインさせる？ Pre-defined セット固定？
  - カードの枚数は有限？無限？
- 自然数：0, +1, *2
  - あるいは、0～255の全ての定数カード？
- 基本的な関数：S K I
  - http://en.wikipedia.org/wiki/SKI_combinator_calculus
  - もっと違う基本関数セット？

議論：2月～ 5月

- どのようなカードがあるべきか？
  - プレイヤーに独自カードをデザインさせる？ Pre-defined セット固定？
  - カードの枚数は有限？無限？

➔ 参加のハードルを低くできた方がよい
  - そんなに複雑にしなくても十分難しい

議論：2月～5月

- 自然数
  - 0, +1, *2 ？
  - 0 ～ 255 の全ての定数カード？

- かなり後半まで議論の対象
  - 全部用意した方が簡単だが
  - +1, *2 で自然数作るくらいはさほど難しくない
  - “ver. 3” まで持ち越し

SKI コンビネータ

- I x = x
- K x y = x
- S x y z = x z (y z)

議論：ＳＫＩは弱すぎないか？

- こんな変態言語で制御構造を
  すべて表現しろ、という課題は難しすぎないか？
- もっと便利な関数をカードとして用意すべき？

- 結論： SKIでコーディングするのは意外にも簡単
- 問題ない
  - S(S(dec)(dec))(dec) = fun x => “相手の s_x に3ダメージ”
  - S(get)(I)(0) = 無限ループ

議論 ：ＳＫＩは強力すぎないか？

- チューリング完全
  - 無限に dec するループが書ける
  - 十数ターンで 2^65556 回 dec を打つ関数が書ける
  - このくらいのターンで決着がつくゲームに収束してしまう？
- 代案：
  - 「多項式時間計算可能な関数しか書けないλ計算のフラグメント」をカード化する

議論 ：ＳＫＩは強力すぎないか？

- 代案 ：
  - 「多項式時間計算可能な関数しか書けないλ計算のフラグメント」をカード化する
- 結論 ：
  - そのλ計算は難しすぎる
  - そもそも、多大な計算を行う関数を短く書く勝負、が面白いポイントであるはず。
  - 単に、１ターンあたりの関数適用の回数に上限を設ける（最終版：1000回）

5月14日：ゲームルールと並行して…

ユーザーのプログラム提出方式確定&公開

- 「64bit Debian GNU/Linux で動く実行ファイルを提出」

ICFPコンテストの問題に要求されること

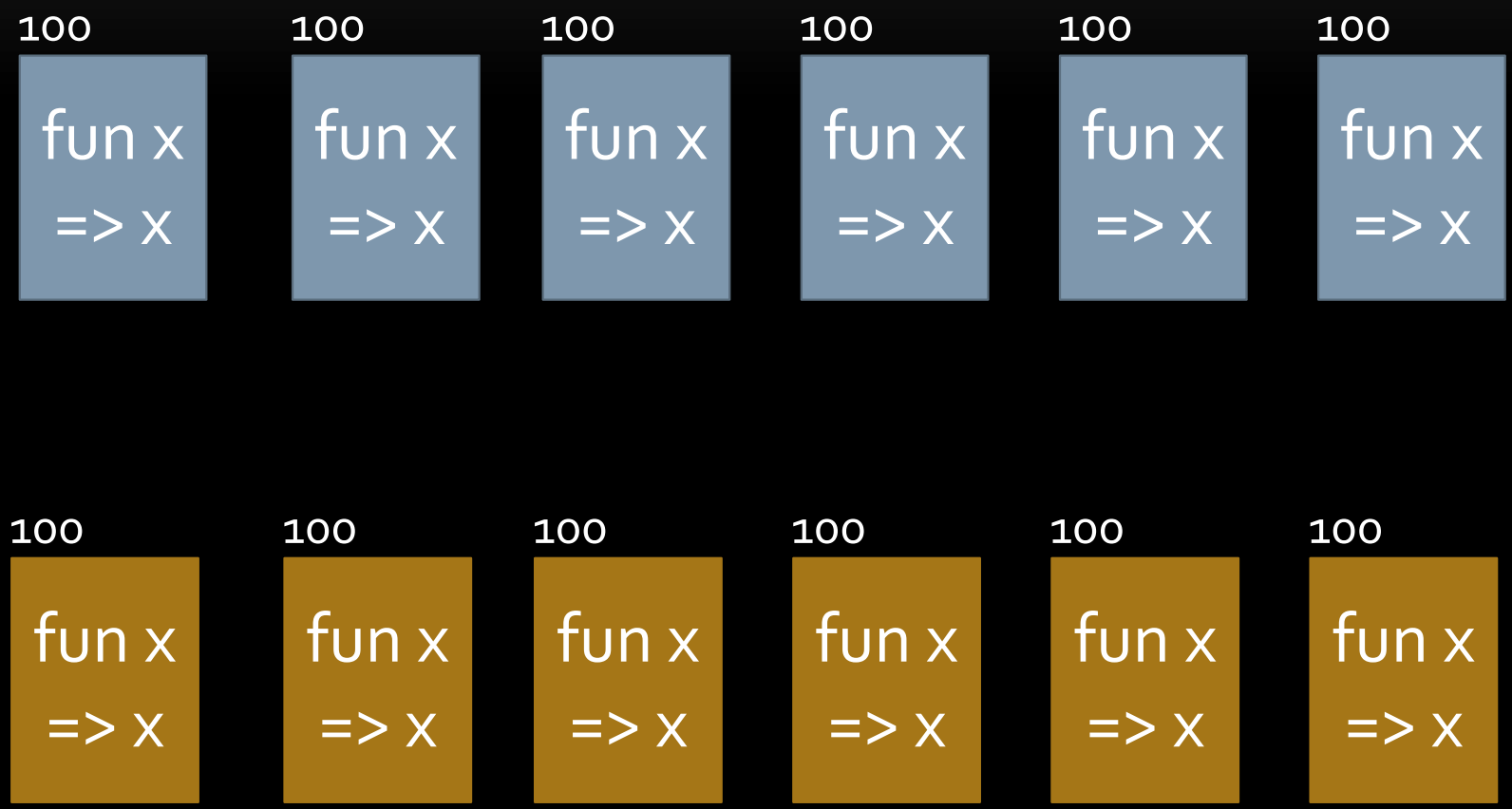
- 可能な限り多彩な言語で参加できること
- 参加のハードルを低くできた方がよい

- 過去の例
  - 簡易VMの仕様と、その上で動くバイナリとしてシステムを提供 06,07,08,09
  - 参加者のマシンで実行した最適化の結果出力 03, 10
  - 独自の低級言語のスクリプトを提出してもらう 04
  - Linux に固定。使いたいパッケージを事前/コンテスト中に申請してもらう 05, 11

5月20日

- ver 3. native.ml
  - 最終版に近いカード構成
  - スロットとモンスターを統合

256匹の関数スロット(HP:100)
“やられたスロットの関数は使えない”



カードの変遷

- 相手のスロットをコピーするカード
  - copy : 自殺して代わりに相手をコピー
  - eat : 死んでる相手のスロットをコピー

➔

- copy : 単にコピー
  - 「相手の作ったカードを簡単に丸コピー」は強すぎるので相応の対価が必要…と考えていたが、
    そもそも「他人が組んだルーチンを解析して自分に取り込める」というプレー自体が高度、それ以上難しくしなくてよい。

カードの変遷

- revive
  - 特定スロットを殺して他を生き返す「メガザル」

➔

- revive : ノーコストで HP 1 で生き返し
  - 元々、「初撃が速い方が相手の重要スロットを倒してそのままワンサイドゲーム」という初速だけでゲームが決まるのを避けたかったために導入（たしか）。
  - このくらいコストを下げないとその効果がでない。

カードの変遷

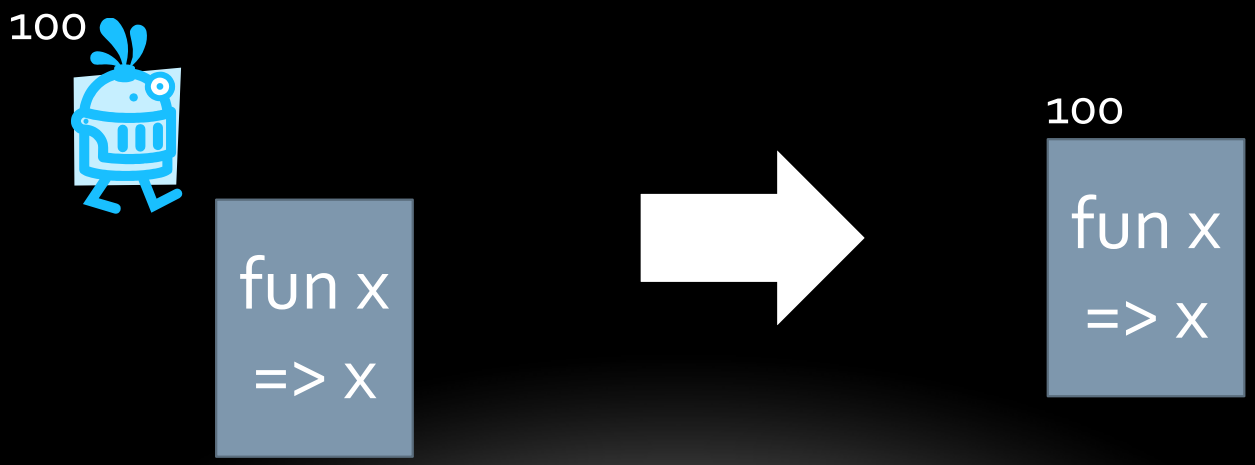
- zombie
  - revive の簡単化に伴い導入
  - 相手の死んだスロットを自分の関数で書きつぶす
    - reviveが簡単すぎるとスロットを倒すメリットが減ってしまったため、相手の重要な砲台をつぶす手段として導入

➔

- zombie
  - 書きつぶす + 相手ターンに相手権限で実行
    - もっと派手にゾンビが暴れられた方が戦略が増えて面白い

カードの変遷

- スロットとモンスターが連携
  - 「何番目」を倒すかの違いが重要



カードの変遷

- 自然数
  - 0, +1, *2 ？
  - 0 ～ 255 の全ての定数カード？
- 作りやすい番号と作りにくい番号で傾斜をつけたほうが面白い
- さらに攻撃と防御で変化
  - inc i = “自分の第 i スロットのHP +=1”
  - dec i = “相手のの第 255-i スロットHP -=1”

議論：ゲームバランス

- 関数適用数上限 ➔ 1ターンあたりのダメージ上限
- スロットの HP の上限
- 引き分けと見なすターン数の上限
  - のバランス

- 遅い単純なルーチンのターン数
- 最高速で敵を殲滅するルーチンのターン数
- 複雑なギミック（ゾンビなど）構築までの必要ターン数
  - を数えて調整

最終版のルール

Lambda: the Gathering

15種類のカード : 自然数

- 定数 0
- +1
- *2

15種類のカード : SKIコンビネータ

- S x y z = x z (y z)
- K x y = x
- I x = x

15種類のカード : 単純攻撃・回復

- inc i = “自分の第 i スロットのHP +=1”
- dec i = “相手のの第 255-i スロットHP -=1”

15種類のカード : 自己犠牲攻撃・回復

- help = “自分に n ダメージかつ味方を 1.1n 回復”
- attack = “自分に n ダメージ、相手に 0.9n ダメージ”

15種類のカード : カードコピー系

- get = “自分の第 i スロットをロード”
- copy = “相手の第 i スロットをロード”
- put x y = y （スロット初期化カード）

15種類のカード

- revive =
  “生命力≦0の味方を1で生き返す”

Lambda: the Gathering

- zombie = “生命力 o の相手のスロットに関数を書き込んで「ゾンビ」化する”
  - ゾンビは相手の次ターン頭に攻撃/回復の符号を反転して１ターンだけ活動し、I に戻る

最終版のパラメタ

- スロット 256 個
- 初期 HP 10000, 最大 65536
- 相手を全滅させれば勝ち (勝ち点 6)
- 先・後 各10万ターン終了時に
  生存スロットが多い方が勝ち (勝ち点 2)
- 各ターンは 1000 回の関数適用で強制終了

Systems

システム・マシン環境

Contest Website

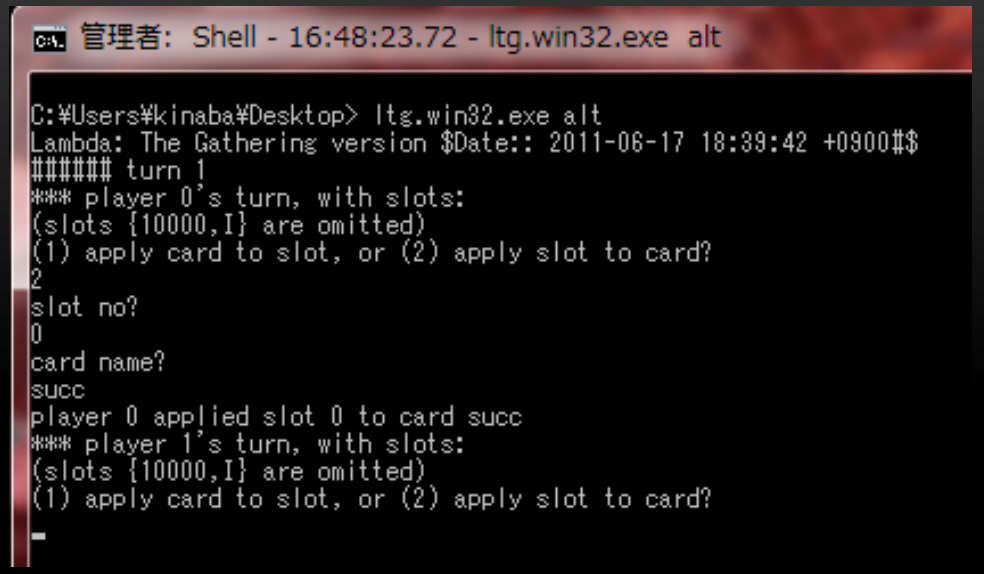
コンテストのサイトに要求されること

- 見られればよい
- できるだけ手間がかからない方がよい

Website : Blogger (無料Blogサービス) を利用

- アナウンス
- 問題文
  - すべてブログ記事
- 質問・返答
  - コメント欄 (Moderated)

ゲーム審判プログラム



- コンテスト開始と同時に公開
  - CUI で人間も遊べる
  - stdin/outでつないでプログラム対戦
  - 実際の順位決定戦にもそのまま使用
- OCaml で実装
  - Windows, MacOS X, Linux 版を提供

“非公式” 対戦サーバ

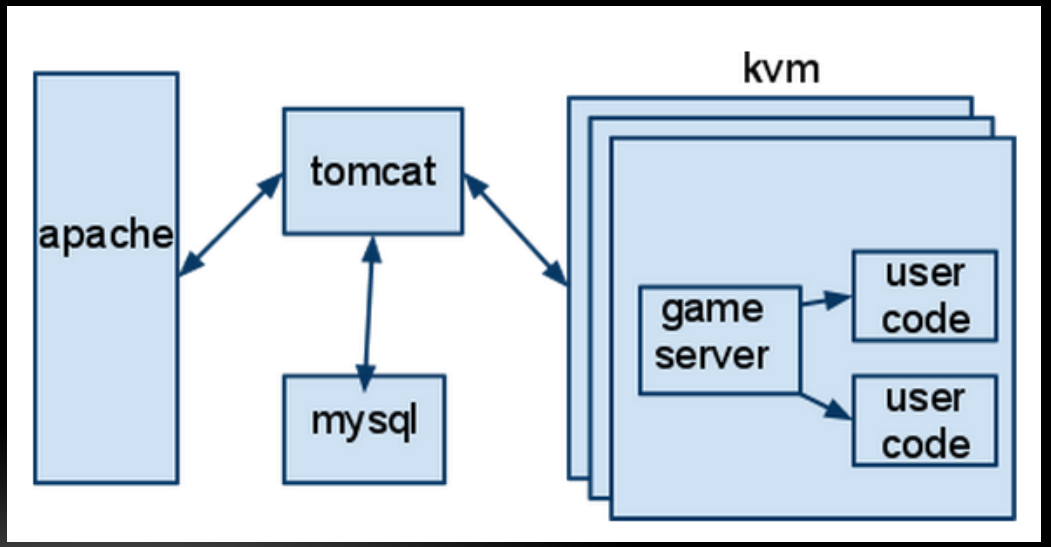
参加者の登録したルーチンをランダム対戦

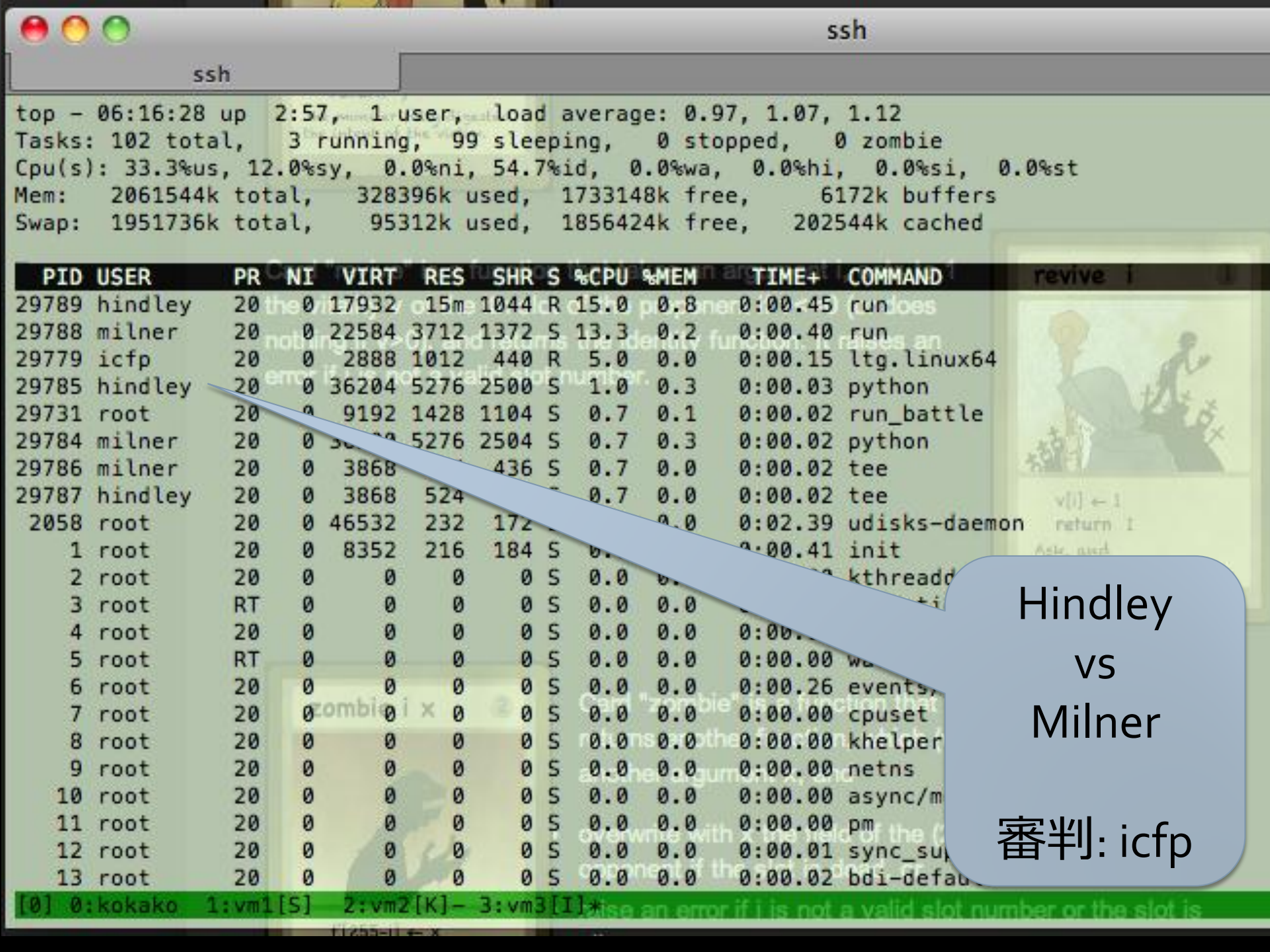
要求されること

- 各チームが ”自分のどのくらいよくやっているか” わかること。
  - 上位チームが切磋琢磨できるように。
  - 少人数チームが不利にならないように。
- サンドボックス
  - ネットワークを遮断。暴走ルーチンをちゃんと殺す

“非公式” 対戦サーバ：マシン環境

- 4コアのワークステーション１台
  - フロントエンド apache + tomcat
  - 3台の kvm で対戦




ssh
ssh
top - 06:16:28 up 2:57, 1 user, load average: 0.97, 1.07, 1.12
Tasks: 102 total, 3 running, 99 sleeping, 0 stopped, 0 zombie
Cpu(s): 33.3%us, 12.0%sy, 0.0%ni, 54.7%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 2061544k total, 328396k used, 1733148k free, 6172k buffers
Swap: 1951736k total, 95312k used, 1856424k free, 202544k cached
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
29789 hindley 20 0 17932 15m 1044 R 15.0 0.8 0:00.45 run
29788 milner 20 0 22584 3712 1372 S 13.3 0.2 0:00.40 run
29779 icfp 20 0 2888 1012 440 R 5.0 0.0 0:00.15 ltg.linux64
29785 hindley 20 0 36204 5276 2500 S 1.0 0.3 0:00.03 python
29731 root 20 0 9192 1428 1104 S 0.7 0.1 0:00.02 run_battle
29784 milner 20 0 5276 2504 S 0.7 0.3 0:00.02 python
29786 milner 20 0 3868 436 S 0.7 0.0 0:00.02 tee
29787 hindley 20 0 3868 524 0.7 0.0 0:00.02 tee
2058 root 20 0 46532 232 172 0.0 0:02.39 udisks-daemon
1 root 20 0 8352 216 184 S 0.0 0:00.41 init
2 root 20 0 0 0 0 S 0.0
3 root RT 0 0 0 0 S 0.0 0.0
4 root 20 0 0 0 0 S 0.0 0.0 0:00.
5 root RT 0 0 0 0 S 0.0 0.0 0:00.00
6 root 20 0 0 0 0 S 0.0 0.0 0:00.26 events/
7 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuset
8 root 20 0 0 0 0 S 0.0 0.0 0:00.00 khelper
9 root 20 0 0 0 0 S 0.0 0.0 0:00.00 netns
10 root 20 0 0 0 0 S 0.0 0.0 0:00.00 async/m
11 root 20 0 0 0 0 S 0.0 0.0 0:00.00 pm
12 root 20 0 0 0 0 S 0.0 0.0 0:00.01 sync_su
13 root 20 0 0 0 0 S 0.0 0.0 0:00.02 bdi-defau
[0] 0:kokako 1:vm1[S] 2:vm2[K]- 3:vm3[I]*
Hindley
vs
Milner
審判: icfp

“非公式” 対戦サーバ : 性能

- 約 8 戦 / 分
- 登録チーム: 約300チーム
  - 1チームあたり、時速 3.3 戦

- 満足の行く回転率にはほど遠い
  - ➔「上位30チームは上位同士で対戦」する
    プレミアリーグ制で workaround

公式の審判環境

- 期間終了後、提出されたプログラムで
  リーグ戦を組んで順位決定

要求

- 優勝者に、本会議（3ヶ月後）に
  間に合うタイミングで招待状を出せること

公式の審判環境 : InTrigger

- InTrigger クラスタ
  - http://intrigger.jp/

審判にかかる時間

- 想定: 300 team
  - 総当たり (先後) 300*300 ゲーム
    - 各 100000 ターン
    - 持ち時間は 各10000秒
  - ≒ 2万日 / 並列度

順位決定方法

- 時間短縮のためのプラン
  - 全チームで スイス式トーナメント
    - ➔結果的に総当たりで十分だったので総当たり（ほとんどのチームがノータイムで）
    - ➔“非公式”サーバが本番より厳しい時間制約を課していたため？
  - 上位30チーム総当たり

Results & Stats

コンテストの結果

Final Standings

1.	**Eta-LOng Normal Form**	**F#**
2.	**Unagi: The Gathering**	**Shell Script and C++**
3.	atomic $ save Madoka	
4.	Punch Jordan in the Face	
5.	**Frictionless Bananas**	**(best 1-person team)**
6.	shinh3	
7.	Wile E.	
8.	Wolf	
9.	Joho	
10.	The Invisible Imp	

(stats from the official slide)

Country Statistics (1.0 per team)			
丄	42	The Netherlands	4
USA	27.32	Canada	2.5
Russia	27	Denmark	2.46
Japan	26.38	Australia	2.33
Ukraine	12.08	South Africa	2.33
France	11.1	Belarus	2
Germany	9.58	Italy	2
UK	7.33	Latvia	1.5

43/45

概ね近年の
傾向通り。
US, JP, RU 3強

(stats from the official slide)

Language Statistics (1.0/team)			
Haskell	27.58	C#	6.33
Python	27.53	F#	6
C++	22.95	Scheme	4.67
OCaml	20.17	Scala	4.03
上	20	Perl	3.62
Java	14.67	Lisp	2.83
Ruby	9.67	Bash	2.17
C	7.67	SML	2

44/45

概ね近年の
傾向通り。
Haskell, C++,
Python, OCaml,
Java。

Thank you for listening!

- ICFP Programming Contest
  - http://icfpcontest.org/
- Wikipedia
  - http://en.wikipedia.org/wiki/ICFP_Programming_Contest
- Yet Another Unofficial Judge for ICFP Contest 2011
  - http://www.paraiso-lang.org/Walpurgisnacht/index.cgi
- 感想リンク集
  - 2011: http://gulfweed.starlancer.org/?ICFP%202011%20write%20ups
  - 2010: http://gulfweed.starlancer.org/?ICFP%202010%20write%20ups
  - 2009: http://d.hatena.ne.jp/hogelog/20090701/p1
  - 2007: http://route477.net/w/?ICFP2007Links