

# ICFP Programming Contest 2014 – Supermassive Black Hom-set Post-mortem

P. Lepin

# What's up with the name?

- Time pressure – wanted to submit ASAP, needed *some* team name.

# What's up with the name?

- Time pressure – wanted to submit ASAP, needed *some* team name.
- Last year I played as “By Wadler’s Beard!” Should have kept it.

# The Plan

Coding directly in GCC assembly possible, but painful. So...

# The Plan

Coding directly in GCC assembly possible, but painful. So...

- 1 Implement the VM.
- 2 Write a parser for a stand-alone HLL or implement an eDSL.
- 3 Implement static checks and/or optimizations.
- 4 Implement codegen.
- 5 ...?
- 6 **(end goal)** Implement the Lambda-Man AI.

# The Plan

Coding directly in GCC assembly possible, but painful. So...

- 1 ~~Implement the VM.~~ – there was a web-based implementation
- 2 Write a parser for a stand-alone HLL ~~or implement an eDSL.~~
- 3 ~~Implement static checks and/or optimizations.~~
- 4 Implement codegen.
- 5 ...?
- 6 **(end goal)** Implement the Lambda-Man AI.

# The Plan

Coding directly in GCC assembly possible, but painful. So...

- ❶ ~~Implement the VM.~~ – there was a web-based implementation
- ❷ Write a parser for a stand-alone HLL ~~or implement an eDSL.~~
- ❸ ~~Implement static checks and/or optimizations.~~
- ❹ Implement codegen.
- ❺ ...?
- ❻ **(end goal)** Implement the Lambda-Man AI.
- ❼ Implement symbolic labels on top of GHC assembly.
- ❽ **(end goal)** Implement a Ghost AI (or several).

# The Plan

Coding directly in GCC assembly possible, but painful. So...

- ❶ ~~Implement the VM.~~ – there was a web-based implementation
- ❷ Write a parser for a stand-alone HLL ~~or implement an eDSL.~~
- ❸ ~~Implement static checks and/or optimizations.~~
- ❹ Implement codegen.
- ❺ ...?
- ❻ **(end goal)** Implement the Lambda-Man AI.
- ❼ Implement symbolic labels on top of GHC assembly.
- ❽ **(end goal)** Implement a Ghost AI (or several).

...some of these decisions were extremely myopic.



# Perceived Fun Factor

	Tools	AI
Lambda-Man	<b>FUN!</b>	<b>FUN!</b>
Ghosts	Less fun.	Less fun.

Lisp-machine CPU, compilers, fairly sophisticated AIs – interesting. 8-bit CPUs and severely resource-constrained programs – not so much.

# Effort

	Tools	AI
Lambda-Man	~ <b>9 hrs</b> , 205 sloc	~ <b>25 hrs</b> , 654 sloc, 3114 instructions compiled
Ghosts	~ <b>3 hrs</b> , 259 sloc	~ <b>3 hrs</b> , 91 instruction

# Wild Guesstimate Of Impact

	Tools	AI
Lambda-Man	Some.	
Ghosts	<b>HUGE!</b>	

Judging by the results of home-brewed tournaments on reddit.

# HLL Compiler Targetting GCC

- HLL is Lisp-like, mimicking Scheme and Clojure.

# HLL Compiler Targetting GCC

- HLL is Lisp-like, mimicking Scheme and Clojure.

```
467 (def foldl
468     (fun [f init xs]
469         (if [atom? xs]
470             init
471             (recur f (f init (car xs)) (cdr xs)))))
472 (def map-map (fun [f xs] (map (fun [x] (cons x (f x))) xs)))
473 (def map
474     (fun [f xs]
475         (if [atom? xs]
476             NIL
477             (cons (f (car xs)) (map f (cdr xs))))))
```

Figure: Almost looks like the real thing.

# HLL Compiler Targetting GCC

- HLL is Lisp-like, mimicking Scheme and Clojure.
- Almost purely functional, `set!` is accepted by the parser, but...

# HLL Compiler Targetting GCC

- HLL is Lisp-like, mimicking Scheme and Clojure.
- Almost purely functional, `set!` is accepted by the parser, but...

```
221 cg n env labels (Spec "!1" [Lit x]) = ([Op "LD" [Num x, Num 1]], labels)
222 -- Unsupported: set!
223 cg n env labels expr = (unsafePerformIO $ hPutStrLn stderr $ "WARNING!!! Unable
```

Figure: Not really supported.

# HLL Compiler Targetting GCC

- HLL is Lisp-like, mimicking Scheme and Clojure.
- Almost purely functional, `set!` is accepted by the parser, but... is not supported.
- The only effectful part is `do/debug`.



# HLL Compiler Targetting GCC

- HLL is Lisp-like, mimicking Scheme and Clojure.
- Almost purely functional, `set!` is accepted by the parser, but... is not supported.
- The only effectful part is `do/debug`.
- **Many** quirks:

# HLL Compiler Targetting GCC

- HLL is Lisp-like, mimicking Scheme and Clojure.
- Almost purely functional, `set!` is accepted by the parser, but... is not supported.
- The only effectful part is `do/debug`.
- **Many** quirks:
  - No general TCE, explicit (and unchecked) tail recursion optimization using `recur`.

# HLL Compiler Targetting GCC

- HLL is Lisp-like, mimicking Scheme and Clojure.
- Almost purely functional, `set!` is accepted by the parser, but... is not supported.
- The only effectful part is `do/debug`.
- **Many** quirks:
  - No general TCE, explicit (and unchecked) tail recursion optimization using `recur`.
  - Mildly insane function call convention to support `recur` – incompatible with ABI as in spec.

# HLL Compiler Targetting GCC

- HLL is Lisp-like, mimicking Scheme and Clojure.
- Almost purely functional, `set!` is accepted by the parser, but... is not supported.
- The only effectful part is `do/debug`.
- **Many** quirks:
  - No general TCE, explicit (and unchecked) tail recursion optimization using `recur`.
  - Mildly insane function call convention to support `recur` – incompatible with ABI as in spec.
  - `ifs` must be in a tail position - no support for `SEL`.

# HLL Compiler Targetting GCC

- HLL is Lisp-like, mimicking Scheme and Clojure.
- Almost purely functional, `set!` is accepted by the parser, but... is not supported.
- The only effectful part is `do/debug`.
- **Many** quirks:
  - No general TCE, explicit (and unchecked) tail recursion optimization using `recur`.
  - Mildly insane function call convention to support `recur` – incompatible with ABI as in spec.
  - `ifs` must be in a tail position - no support for `SEL`.
  - Built-ins such as `car` or `+` are special forms rather than first-class entities.

# HLL Compiler Targetting GCC

- HLL is Lisp-like, mimicking Scheme and Clojure.
- Almost purely functional, `set!` is accepted by the parser, but... is not supported.
- The only effectful part is `do/debug`.
- **Many** quirks:
  - No general TCE, explicit (and unchecked) tail recursion optimization using `recur`.
  - Mildly insane function call convention to support `recur` – incompatible with ABI as in spec.
  - `ifs` must be in a tail position - no support for `SEL`.
  - Built-ins such as `car` or `+` are special forms rather than first-class entities.
  - No macros.

# HLL Compiler Targetting GCC

- HLL is Lisp-like, mimicking Scheme and Clojure.
- Almost purely functional, `set!` is accepted by the parser, but... is not supported.
- The only effectful part is `do/debug`.
- **Many** quirks:
  - No general TCE, explicit (and unchecked) tail recursion optimization using `recur`.
  - Mildly insane function call convention to support `recur` – incompatible with ABI as in spec.
  - `ifs` must be in a tail position - no support for `SEL`.
  - Built-ins such as `car` or `+` are special forms rather than first-class entities.
  - No macros.
  - Virtually no diagnostics or static checks – a pain to debug.

# HLL Compiler Targetting GCC

- HLL is Lisp-like, mimicking Scheme and Clojure.
- Almost purely functional, `set!` is accepted by the parser, but... is not supported.
- The only effectful part is `do/debug`.
- **Many** quirks:
  - No general TCE, explicit (and unchecked) tail recursion optimization using `recur`.
  - Mildly insane function call convention to support `recur` – incompatible with ABI as in spec.
  - `ifs` must be in a tail position - no support for `SEL`.
  - Built-ins such as `car` or `+` are special forms rather than first-class entities.
  - No macros.
  - Virtually no diagnostics or static checks – a pain to debug.
- Implementing as eDSL could have given some type safety *almost for free*.



# HLL Compiler Targetting GCC – Bugs

- A couple of nasty bugs ~30 hrs into the contest.

# HLL Compiler Targetting GCC – Bugs

- A couple of nasty bugs ~30 hrs into the contest.

212	212	cg n env labels (Spec "do" []) = ([], labels)
213		-cg n env labels (Spec "do" (expr : rest)) = (code' ++ code'', labels')
	213	+cg n env labels (Spec "do" (expr : rest)) = (code' ++ code'', labels'!)
214	214	where

**Figure:** The fact that it typechecks doesn't *always* mean it's correct. (But encoding the codegen in a monad *could* have helped.)

# HLL Compiler Targetting GCC – Bugs

- A couple of nasty bugs ~30 hrs into the contest.

212	212	cg n env labels (Spec "do" []) = ([], labels)
213		-cg n env labels (Spec "do" (expr : rest)) = (code' ++ code'', labels')
	213	+cg n env labels (Spec "do" (expr : rest)) = (code' ++ code'', labels'!)
214	214	where

**Figure:** The fact that it typechecks doesn't *always* mean it's correct. (But encoding the codegen in a monad *could* have helped.)

208	208	-- In general, never apply to unnamed functions: might be expensive
209		-cg n env labels (Spec "recur" args) = ([Op "LD" [Num 0, Num 0]] ++ code' ++ [Op "LD" [Num 0,
	209	+cg n env labels (Spec "recur" args) = ([Op "LD" ("@recur" `var_lookup` (n, env))] ++ code'
210	210	where

**Figure:** recur only worked – occasionally – by accident.

# Lambda-Man AI

- Lightning round AI retained as a fall-back – main AI may not return a move in situations it considers hopeless:
  - Manhattan distance to the closest pill as a value function.
  - Doesn't like being too close to ghosts or visiting recently seen locations (reduces the effect of local minima).

# Lambda-Man AI

- Lightning round AI retained as a fall-back – main AI may not return a move in situations it considers hopeless:
  - Manhattan distance to the closest pill as a value function.
  - Doesn't like being too close to ghosts or visiting recently seen locations (reduces the effect of local minima).
- BFS: cuts off on dying, reaching anything edible or exceeding the depth limit.

# Lambda-Man AI

- Lightning round AI retained as a fall-back – main AI may not return a move in situations it considers hopeless:
  - Manhattan distance to the closest pill as a value function.
  - Doesn't like being too close to ghosts or visiting recently seen locations (reduces the effect of local minima).
- BFS: cuts off on dying, reaching anything edible or exceeding the depth limit.
  - Being close to ghosts is penalized.

# Lambda-Man AI

- Lightning round AI retained as a fall-back – main AI may not return a move in situations it considers hopeless:
  - Manhattan distance to the closest pill as a value function.
  - Doesn't like being too close to ghosts or visiting recently seen locations (reduces the effect of local minima).
- BFS: cuts off on dying, reaching anything edible or exceeding the depth limit.
  - Being close to ghosts is penalized.
  - If nothing edible found, simple heuristic value function kicks in.

# Lambda-Man AI

- Lightning round AI retained as a fall-back – main AI may not return a move in situations it considers hopeless:
  - Manhattan distance to the closest pill as a value function.
  - Doesn't like being too close to ghosts or visiting recently seen locations (reduces the effect of local minima).
- BFS: cuts off on dying, reaching anything edible or exceeding the depth limit.
  - Being close to ghosts is penalized.
  - If nothing edible found, simple heuristic value function kicks in.
  - Likes running towards nearest power pill when ghosts are nearby.



# Lambda-Man AI

- Lightning round AI retained as a fall-back – main AI may not return a move in situations it considers hopeless:
  - Manhattan distance to the closest pill as a value function.
  - Doesn't like being too close to ghosts or visiting recently seen locations (reduces the effect of local minima).
- BFS: cuts off on dying, reaching anything edible or exceeding the depth limit.
- Data structures: binary search trees (logarithmic access map and IntSet), simple queue from *PFDS*.

# Lambda-Man AI

- Lightning round AI retained as a fall-back – main AI may not return a move in situations it considers hopeless:
  - Manhattan distance to the closest pill as a value function.
  - Doesn't like being too close to ghosts or visiting recently seen locations (reduces the effect of local minima).
- BFS: cuts off on dying, reaching anything edible or exceeding the depth limit.
- Data structures: binary search trees (logarithmic access map and IntSet), simple queue from *PFDS*.

```
404 (def q-append
405   (fun [q xs]
406     (q-norm (cons (car q) (foldl (fun [r x] (cons x r)) (cdr q) xs)))))
407 (def q-norm
408   (fun [q]
409     (if [atom? (car q)]
410       (cons (reverse (cdr q)) NIL)
411       q)))
```

# Lambda-Man AI

- Lightning round AI retained as a fall-back – main AI may not return a move in situations it considers hopeless:
  - Manhattan distance to the closest pill as a value function.
  - Doesn't like being too close to ghosts or visiting recently seen locations (reduces the effect of local minima).
- BFS: cuts off on dying, reaching anything edible or exceeding the depth limit.
- Data structures: binary search trees (logarithmic access map and IntSet), simple queue from *PFDS*.
- Propagates ghosts (ignoring differences in speed) as long as they have no choice.

# Lambda-Man AI

- Lightning round AI retained as a fall-back – main AI may not return a move in situations it considers hopeless:
  - Manhattan distance to the closest pill as a value function.
  - Doesn't like being too close to ghosts or visiting recently seen locations (reduces the effect of local minima).
- BFS: cuts off on dying, reaching anything edible or exceeding the depth limit.
- Data structures: binary search trees (logarithmic access map and IntSet), simple queue from *PFDS*.
- Propagates ghosts (ignoring differences in speed) as long as they have no choice.



Figure: Initial state.

# Lambda-Man AI

- Lightning round AI retained as a fall-back – main AI may not return a move in situations it considers hopeless:
  - Manhattan distance to the closest pill as a value function.
  - Doesn't like being too close to ghosts or visiting recently seen locations (reduces the effect of local minima).
- BFS: cuts off on dying, reaching anything edible or exceeding the depth limit.
- Data structures: binary search trees (logarithmic access map and IntSet), simple queue from *PFDS*.
- Propagates ghosts (ignoring differences in speed) as long as they have no choice.

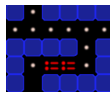


Figure: After 1 step.

# Lambda-Man AI

- Lightning round AI retained as a fall-back – main AI may not return a move in situations it considers hopeless:
  - Manhattan distance to the closest pill as a value function.
  - Doesn't like being too close to ghosts or visiting recently seen locations (reduces the effect of local minima).
- BFS: cuts off on dying, reaching anything edible or exceeding the depth limit.
- Data structures: binary search trees (logarithmic access map and IntSet), simple queue from *PFDS*.
- Propagates ghosts (ignoring differences in speed) as long as they have no choice.



Figure: After 2 steps.

# Lambda-Man AI

- Lightning round AI retained as a fall-back – main AI may not return a move in situations it considers hopeless:
  - Manhattan distance to the closest pill as a value function.
  - Doesn't like being too close to ghosts or visiting recently seen locations (reduces the effect of local minima).
- BFS: cuts off on dying, reaching anything edible or exceeding the depth limit.
- Data structures: binary search trees (logarithmic access map and IntSet), simple queue from *PFDS*.
- Propagates ghosts (ignoring differences in speed) as long as they have no choice.



Figure: After 3 steps.

# Lambda-Man AI

- Lightning round AI retained as a fall-back – main AI may not return a move in situations it considers hopeless:
  - Manhattan distance to the closest pill as a value function.
  - Doesn't like being too close to ghosts or visiting recently seen locations (reduces the effect of local minima).
- BFS: cuts off on dying, reaching anything edible or exceeding the depth limit.
- Data structures: binary search trees (logarithmic access map and IntSet), simple queue from *PFDS*.
- Propagates ghosts (ignoring differences in speed) as long as they have no choice.



Figure: After 4 or more steps.



# Lambda-Man AI

- Lightning round AI retained as a fall-back – main AI may not return a move in situations it considers hopeless:
  - Manhattan distance to the closest pill as a value function.
  - Doesn't like being too close to ghosts or visiting recently seen locations (reduces the effect of local minima).
- BFS: cuts off on dying, reaching anything edible or exceeding the depth limit.
- Data structures: binary search trees (logarithmic access map and IntSet), simple queue from *PFDS*.
- Propagates ghosts (ignoring differences in speed) as long as they have no choice.
- Value of scared ghosts and fruits discounted by time to expiration.

# Lambda-Man AI – Quirks

- Termination of search on reaching anything edible leads to “interesting” behavior.

# Lambda-Man AI – Quirks

- Termination of search on reaching anything edible leads to “interesting” behavior.
- Scaredy Lambda-Man – pessimistic estimates of ghost movement.

# Lambda-Man AI – Quirks

- Termination of search on reaching anything edible leads to “interesting” behavior.
- Scaredy Lambda-Man – pessimistic estimates of ghost movement.
- Very little global preprocessing – computing tunnels as graph edges was tempting.

# Lambda-Man AI – Quirks

- Termination of search on reaching anything edible leads to “interesting” behavior.
- Scaredy Lambda-Man – pessimistic estimates of ghost movement.
- Very little global preprocessing – computing tunnels as graph edges was tempting.
- Unnecessary map preprocessing on each step.

# Lambda-Man AI – Quirks

- Termination of search on reaching anything edible leads to “interesting” behavior.
- Scaredy Lambda-Man – pessimistic estimates of ghost movement.
- Very little global preprocessing – computing tunnels as graph edges was tempting.
- Unnecessary map preprocessing on each step.
- AI state fragile – failures screw up future behavior.

# Lambda-Man AI – Quirks

- Termination of search on reaching anything edible leads to “interesting” behavior.
- Scaredy Lambda-Man – pessimistic estimates of ghost movement.
- Very little global preprocessing – computing tunnels as graph edges was tempting.
- Unnecessary map preprocessing on each step.
- AI state fragile – failures screw up future behavior.
- **Not-a-quirk:** Ghost AIs known, but emulation impractical due to cycle limit.

# Lambda-Man AI – Panic

- A few hours before the deadline...



# Lambda-Man AI – Panic

- A few hours before the deadline...

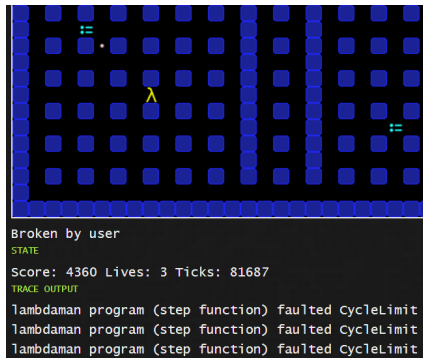


Figure: OMG!!! Cycle limit!

# Lambda-Man AI – Panic

- A few hours before the deadline...
- Hard to estimate cycles spent sanely from inside the simulation.

# Lambda-Man AI – Panic

- A few hours before the deadline...
- Hard to estimate cycles spent sanely from inside the simulation.
- Regretted not having my own VM – web implementation sluggish *and* read-only.

- A few hours before the deadline...
- Hard to estimate cycles spent sanely from inside the simulation.
- Regretted not having my own VM – web implementation sluggish *and* read-only.

```

func main() {
    h$g54b0() {var a=h$g1.d1;h$bh()::h$g12(a,h$mainZOWorldziTypeszscore);return h$mainZOWorldzi
    function h$g546() {var a=h$g1.d1;h$bh()::h$pl(h$g547);return h$e(a)}function h$g545() {va a=h$g1.d1
    function h$g54d() {f--h$sp;h$g1=h$g2(h$mainZCECSDziInterpreterziCons_con_e.h$g1,h$g12);return h$g54c
    function h$g55() {f--h$sp;if(3==h$g1.f.a)h$g1=h$mainZOWorldziPacmanziFruit;else return h$e(h$main
    function h$g55h() {f--h$sp;if(3==h$g1.f.a)h$g1=h$mainZOWorldziPacmanziFruit;else return h$e(h$main
    function h$g550() {var a=h$g54ack[h$sp-1];h$sp--2;switch(h$g1.f.a){case 1:h$g1=h$mainZOWorldziPacma
    function h$g55m() {var a=h$g54ack[h$sp-14],b=h$g54ack[h$sp-13],c=h$g54ack[h$sp-12],d=h$g54ack[h$sp-11]
    function h$g55l() {var a=h$g54ack[h$sp-13],b=h$g54ack[h$sp-12],c=h$g54ack[h$sp-11],d=h$g54ack[h$sp-10]
    function h$mainZOWorldziPacmanziConvertMap_e() {h$pl(h$g55k);return h$e(h$g2)}function h$g55n() {f--
    function h$mainZOWorldziPacmanziCodeCell_e() {h$pl(h$g55n);return h$e(h$g2)}function h$mainZOWor
    function h$g55q() {f--h$sp;h$g1=h$g2(h$mainZCECSDziInterpreterziCons_con_e.h$g1,h$g12);return h$g54c
    function h$g55t() {var a=h$g54ack[h$sp-15],b=h$g54ack[h$sp-14],c=h$g54ack[h$sp-13],d=h$g54ack[h$sp-12]
    function h$g55s() {var a=h$g54ack[h$sp-15],b=h$g54ack[h$sp-14],c=h$g54ack[h$sp-13],d=h$g54ack[h$sp-12]
    function h$g55r() {var a=h$g54ack[h$sp-1];h$sp--2;var b=h$g1.d2,h$pl6(a,h$g1.d1,b,d1,b.d2,b.d3,b.d4
    function h$g55x() {var a=h$g54ack[h$sp-15],b=h$g54ack[h$sp-14],c=h$g54ack[h$sp-13],d=h$g54ack[h$sp-12]
    function h$g55v() {var a=h$g54ack[h$sp-15],b=h$g54ack[h$sp-14],c=h$g54ack[h$sp-13],d=h$g54ack[h$sp-12]
    function h$g55w() {var a=h$g54ack[h$sp-1];h$sp--2;var b=h$g1.d2,h$pl6(a,h$g1.d1,b,d1,b.d2,b.d3,b.d4
    function h$g56d() {var a=h$g1.d1,b=h$g1.d2;h$bh()::h$g12(b,a);return h$ap 1 1 fast()}function h$g56c
    function h$g56b() {var a=h$g54ack[h$sp-2],b=h$g54ack[h$sp-1];h$sp--2;if(1==h$g1.f.a) return h$e(a);a
    function h$g559() {f--h$sp;if(1==h$g1.f.a)h$g1=h$g2(h$mainZCGHCziTypesziZMZN;else{var a=h$g(h$g56a
    function h$g56h() {var a=h$g54ack[h$sp-1];h$sp--2;if(32==h$g1.h$g1.h$g1(h$g56i,h$g1(h$g56j,a));els
    function h$g56m() {var a=h$g54ack[h$sp-1];h$sp--2;if(97==h$g1.h$g1.h$g1(h$g56i,h$g1(h$g56o,a));els
    var h$g56Y=h$g56h$g56i$g56j$g56k$g56l$g56m$g56n$g56o$g56p$g56q$g56r$g56s$g56t$g56u$g56v$g56w$g56x$g56y$g56z$g56aa$g56ab$g56ac$g56ad$g56ae$g56af$g56ag$g56ah$g56ai$g56aj$g56ak$g56al$g56am$g56an$g56ao$g56ap$g56aq$g56ar$g56as$g56at$g56au$g56av$g56aw$g56ax$g56ay$g56az$g56ba$g56bb$g56bc$g56bd$g56be$g56bf$g56bg$g56bh$g56bi$g56bj$g56bk$g56bl$g56bm$g56bn$g56bo$g56bp$g56bq$g56br$g56bs$g56bt$g56bu$g56bv$g56bw$g56bx$g56by$g56bz$g56ca$g56cb$g56cc$g56cd$g56ce$g56cf$g56cg$g56ch$g56ci$g56cj$g56ck$g56cl$g56cm$g56cn$g56co$g56cp$g56cq$g56cr$g56cs$g56ct$g56cu$g56cv$g56cw$g56cx$g56cy$g56cz$g56da$g56db$g56dc$g56dd$g56de$g56df$g56dg$g56dh$g56di$g56dj$g56dk$g56dl$g56dm$g56dn$g56do$g56dp$g56dq$g56dr$g56ds$g56dt$g56du$g56dv$g56dw$g56dx$g56dy$g56dz$g56ea$g56eb$g56ec$g56ed$g56ee$g56ef$g56eg$g56eh$g56ei$g56ej$g56ek$g56el$g56em$g56en$g56eo$g56ep$g56eq$g56er$g56es$g56et$g56eu$g56ev$g56ew$g56ex$g56ey$g56ez$g56fa$g56fb$g56fc$g56fd$g56fe$g56ff$g56fg$g56fh$g56fi$g56fj$g56fk$g56fl$g56fm$g56fn$g56fo$g56fp$g56fq$g56fr$g56fs$g56ft$g56fu$g56fv$g56fw$g56fx$g56fy$g56fz$g56ga$g56gb$g56gc$g56gd$g56ge$g56gf$g56gg$g56gh$g56gi$g56gj$g56gk$g56gl$g56gm$g56gn$g56go$g56gp$g56gq$g56gr$g56gs$g56gt$g56gu$g56gv$g56gw$g56gx$g56gy$g56gz$g56ha$g56hb$g56hc$g56hd$g56he$g56hf$g56hg$g56hh$g56hi$g56hj$g56hk$g56hl$g56hm$g56hn$g56ho$g56hp$g56hq$g56hr$g56hs$g56ht$g56hu$g56hv$g56hw$g56hx$g56hy$g56hz$g56ia$g56ib$g56ic$g56id$g56ie$g56if$g56ig$g56ih$g56ii$g56ij$g56ik$g56il$g56im$g56in$g56io$g56ip$g56iq$g56ir$g56is$g56it$g56iu$g56iv$g56iw$g56ix$g56iy$g56iz$g56ja$g56jb$g56jc$g56jd$g56je$g56jf$g56jg$g56jh$g56ji$g56jj$g56jk$g56jl$g56jm$g56jn$g56jo$g56jp$g56jq$g56jr$g56js$g56jt$g56ju$g56jv$g56jw$g56jx$g56jy$g56jz$g56ka$g56kb$g56kc$g56kd$g56ke$g56kf$g56kg$g56kh$g56ki$g56kj$g56kk$g56kl$g56km$g56kn$g56ko$g56kp$g56kq$g56kr$g56ks$g56kt$g56ku$g56kv$g56kw$g56kx$g56ky$g56kz$g56la$g56lb$g56lc$g56ld$g56le$g56lf$g56lg$g56lh$g56li$g56lj$g56lk$g56ll$g56lm$g56ln$g56lo$g56lp$g56lq$g56lr$g56ls$g56lt$g56lu$g56lv$g56lw$g56lx$g56ly$g56lz$g56ma$g56mb$g56mc$g56md$g56me$g56mf$g56mg$g56mh$g56mi$g56mj$g56mk$g56ml$g56mm$g56mn$g56mo$g56mp$g56mq$g56mr$g56ms$g56mt$g56mu$g56mv$g56mw$g56mx$g56my$g56mz$g56na$g56nb$g56nc$g56nd$g56ne$g56nf$g56ng$g56nh$g56ni$g56nj$g56nk$g56nl$g56nm$g56nn$g56no$g56np$g56nq$g56nr$g56ns$g56nt$g56nu$g56nv$g56nw$g56nx$g56ny$g56nz$g56oa$g56ob$g56oc$g56od$g56oe$g56of$g56og$g56oh$g56oi$g56oj$g56ok$g56ol$g56om$g56on$g56oo$g56op$g56oq$g56or$g56os$g56ot$g56ou$g56ov$g56ow$g56ox$g56oy$g56oz$g56pa$g56pb$g56pc$g56pd$g56pe$g56pf$g56pg$g56ph$g56pi$g56pj$g56pk$g56pl$g56pm$g56pn$g56po$g56pp$g56pq$g56pr$g56ps$g56pt$g56pu$g56pv$g56pw$g56px$g56py$g56pz$g56qa$g56qb$g56qc$g56qd$g56qe$g56qf$g56qg$g56qh$g56qi$g56qj$g56qk$g56ql$g56qm$g56qn$g56qo$g56qp$g56qq$g56qr$g56qs$g56qt$g56qu$g56qv$g56qw$g56qx$g56qy$g56qz$g56ra$g56rb$g56rc$g56rd$g56re$g56rf$g56rg$g56rh$g56ri$g56rj$g56rk$g56rl$g56rm$g56rn$g56ro$g56rp$g56rq$g56rr$g56rs$g56rt$g56ru$g56rv$g56rw$g56rx$g56ry$g56rz$g56sa$g56sb$g56sc$g56sd$g56se$g56sf$g56sg$g56sh$g56si$g56sj$g56sk$g56sl$g56sm$g56sn$g56so$g56sp$g56sq$g56sr$g56ss$g56st$g56su$g56sv$g56sw$g56sx$g56sy$g56sz$g56ta$g56tb$g56tc$g56td$g56te$g56tf$g56tg$g56th$g56ti$g56tj$g56tk$g56tl$g56tm$g56tn$g56to$g56tp$g56tq$g56tr$g56ts$g56tt$g56tu$g56tv$g56tw$g56tx$g56ty$g56tz$g56ua$g56ub$g56uc$g56ud$g56ue$g56uf$g56ug$g56uh
```

Figure: Good luck editing *this*.

# Lambda-Man AI – Panic

- A few hours before the deadline...
- Hard to estimate cycles spent sanely from inside the simulation.
- Regretted not having my own VM – web implementation sluggish *and* read-only.
- Regretted not having macros – no easy inlining.

# Lambda-Man AI – Panic

- A few hours before the deadline...
- Hard to estimate cycles spent sanely from inside the simulation.
- Regretted not having my own VM – web implementation sluggish *and* read-only.
- Regretted not having macros – no easy inlining.
- Random BFS depth cutoffs based on the map size.

# Lambda-Man AI – Panic

- A few hours before the deadline...
- Hard to estimate cycles spent sanely from inside the simulation.
- Regretted not having my own VM – web implementation sluggish *and* read-only.
- Regretted not having macros – no easy inlining.
- Random BFS depth cutoffs based on the map size.
- Optimized by hand.

# Lambda-Man AI – Panic

- A few hours before the deadline...
- Hard to estimate cycles spent sanely from inside the simulation.
- Regretted not having my own VM – web implementation sluggish *and* read-only.
- Regretted not having macros – no easy inlining.
- Random BFS depth cutoffs based on the map size.
- Optimized by hand.
  - Non-tail recursive list HOFs – no artificial limitations on stack depth, RTNs are cheaper than reversing the accumulator.

```
473 (def map
474   (fun [f xs]
475     (if [atom? xs]
476         NIL
477         (cons (f (car xs)) (map f (cdr xs))))))
478 ;(def map (fun [f xs] (reverse (map-rev NIL f xs))))
479 ;(def map-rev
480 ;  (fun [acc f xs]
481 ;    (if [atom? xs]
482 ;        acc
483 ;        (recur (cons (f (car xs)) acc) f (cdr xs)))))
```



# Lambda-Man AI – Panic

- A few hours before the deadline...
- Hard to estimate cycles spent sanely from inside the simulation.
- Regretted not having my own VM – web implementation sluggish *and* read-only.
- Regretted not having macros – no easy inlining.
- Random BFS depth cutoffs based on the map size.
- Optimized by hand.
  - Non-tail recursive list HOFs – no artificial limitations on stack depth, RTNs are cheaper than reversing the accumulator.
  - Fused maps and filters in a few places.

# Lambda-Man AI – Panic

- A few hours before the deadline...
- Hard to estimate cycles spent sanely from inside the simulation.
- Regretted not having my own VM – web implementation sluggish *and* read-only.
- Regretted not having macros – no easy inlining.
- Random BFS depth cutoffs based on the map size.
- Optimized by hand.
  - Non-tail recursive list HOFs – no artificial limitations on stack depth, RTNs are cheaper than reversing the accumulator.
  - Fused maps and filters in a few places.
  - Eliminated some unneeded intermediate variables.

# Lambda-Man AI – Panic

- A few hours before the deadline...
- Hard to estimate cycles spent sanely from inside the simulation.
- Regretted not having my own VM – web implementation sluggish *and* read-only.
- Regretted not having macros – no easy inlining.
- Random BFS depth cutoffs based on the map size.
- Optimized by hand.
  - Non-tail recursive list HOFs – no artificial limitations on stack depth, RTNs are cheaper than reversing the accumulator.
  - Fused maps and filters in a few places.
  - Eliminated some unneeded intermediate variables.
- Seems to have worked.

# Ghost AI

- *Very* limited resources – even accounting for no need to make decisions on most runs.

# Ghost AI

- Very limited resources – even accounting for no need to make decisions on most runs.
- Wanted something simple and reasonably robust.

# Ghost AI

- Very limited resources – even accounting for no need to make decisions on most runs.
- Wanted something simple and reasonably robust.
- Tries to minimize  $L_1$  distance to Lambda-Man (*Blinky/Chaser*-style).

# Ghost AI

- Very limited resources – even accounting for no need to make decisions on most runs.
- Wanted something simple and reasonably robust.
- Tries to minimize  $L_1$  distance to Lambda-Man (*Blinky/Chaser*-style).
  - **Problem:** Ghosts tend to clump together.

# Ghost AI

- Very limited resources – even accounting for no need to make decisions on most runs.
- Wanted something simple and reasonably robust.
- Tries to minimize  $L_1$  distance to Lambda-Man (*Blinky/Chaser*-style).
  - **Problem:** Ghosts tend to clump together.
  - **Problem:** Can get stuck in dead ends easily.



# Ghost AI

- Very limited resources – even accounting for no need to make decisions on most runs.
- Wanted something simple and reasonably robust.
- Tries to minimize  $L_1$  distance to Lambda-Man (*Blinky/Chaser*-style).
- Ghost's index affects tie-breaks: different ghosts favour different directions at intersections.

# Ghost AI

- Very limited resources – even accounting for no need to make decisions on most runs.
- Wanted something simple and reasonably robust.
- Tries to minimize  $L_1$  distance to Lambda-Man (*Blinky/Chaser*-style).
- Ghost's index affects tie-breaks: different ghosts favour different directions at intersections.
- Simple counter: after a few actual decisions in a row resulting in horizontal or vertical move, that axis is excluded.

# Ghost AI

- Very limited resources – even accounting for no need to make decisions on most runs.
- Wanted something simple and reasonably robust.
- Tries to minimize  $L_1$  distance to Lambda-Man (*Blinky/Chaser*-style).
- Ghost's index affects tie-breaks: different ghosts favour different directions at intersections.
- Simple counter: after a few actual decisions in a row resulting in horizontal or vertical move, that axis is excluded.
- Pretty efficient on non-pathological maps: ghosts tend to surround the Lambda-Man.

# Sources

- Submission is available on GitHub, link can be found on ICFPC subreddit, along with many interesting submissions and reports from other teams:  
<http://www.reddit.com/r/icfpcontest>

# How To Win The ICFP Programming Contest

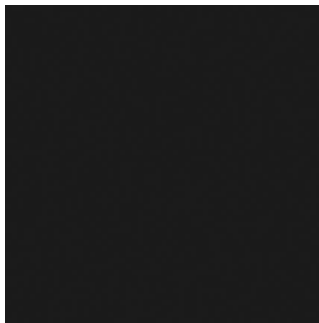
What I learned from past failures and how I used it this time – or not

- Try to decipher the hint.

# How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- Try to decipher the hint.



**Figure:** The apparently black background image on the official site has slight brightness variations. This “steganographic message” turned out to be random noise. D’oh!

# How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~

# How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- Do your research.



# How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- Do your research.
  - There are people smarter than I am on the Net.

# How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- Do your research.
  - There are people smarter than I am on the Net.
  - SECD – little insight into the problem, the spec was enough.

# How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- Do your research.
  - There are people smarter than I am on the Net.
  - SECD – little insight into the problem, the spec was enough.
  - Classic ghost AIs – could have mislead me.

# How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~

# How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- If you want to do well – get on a strong team.

# How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- If you want to do well – get on a strong team.
  - But can be stressful.

# How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- If you want to do well – get on a strong team.
  - But can be stressful.
  - And I'm in it for the sheer fun of it...

# How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- ~~If you want to do well – get on a strong team.~~



# How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- ~~If you want to do well – get on a strong team.~~
- Don't forget to rest, eat and sleep.

# How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- ~~If you want to do well – get on a strong team.~~
- Don't forget to rest, eat and sleep. On the other hand...

# How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- ~~If you want to do well – get on a strong team.~~
- ~~Don't forget to rest, eat and sleep.~~

# How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- ~~If you want to do well – get on a strong team.~~
- ~~Don't forget to rest, eat and sleep.~~
- Random tweaks don't work. Think instead.

# How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- ~~If you want to do well – get on a strong team.~~
- ~~Don't forget to rest, eat and sleep.~~
- Random tweaks don't work. Think instead.

```
[comp-thr (fun [m] (if [<= (bstm-w m) 25] 20 (if [<= (bstm-w m) 35] 12 (if [<= (bstm-w m) 50] 8 5))))]
```

Figure: Cutoff depth selection for BFS. Speaks for itself.

# How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- ~~If you want to do well – get on a strong team.~~
- ~~Don't forget to rest, eat and sleep.~~
- ~~Random tweaks don't work. Think instead.~~

# How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- ~~If you want to do well – get on a strong team.~~
- ~~Don't forget to rest, eat and sleep.~~
- ~~Random tweaks don't work. Think instead.~~
- Read the spec carefully.

# How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- ~~If you want to do well – get on a strong team.~~
- ~~Don't forget to rest, eat and sleep.~~
- ~~Random tweaks don't work. Think instead.~~
- Read the spec carefully.
  - Signed my submissions with SHA256 instead of SHA1.



# How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- ~~If you want to do well – get on a strong team.~~
- ~~Don't forget to rest, eat and sleep.~~
- ~~Random tweaks don't work. Think instead.~~
- Read the spec carefully.
  - Signed my submissions with SHA256 instead of SHA1.
  - Only noticed minutes before the deadline.

# How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- ~~If you want to do well – get on a strong team.~~
- ~~Don't forget to rest, eat and sleep.~~
- ~~Random tweaks don't work. Think instead.~~
- ~~Read the spec carefully.~~

# How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- ~~If you want to do well – get on a strong team.~~
- ~~Don't forget to rest, eat and sleep.~~
- ~~Random tweaks don't work. Think instead.~~
- ~~Read the spec carefully.~~
- Use C++.

# How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- ~~If you want to do well – get on a strong team.~~
- ~~Don't forget to rest, eat and sleep.~~
- ~~Random tweaks don't work. Think instead.~~
- ~~Read the spec carefully.~~
- Use C++.
  - It appears to dominate the field as far as the tools of choice for discriminating hackers are concerned.

# How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- ~~If you want to do well – get on a strong team.~~
- ~~Don't forget to rest, eat and sleep.~~
- ~~Random tweaks don't work. Think instead.~~
- ~~Read the spec carefully.~~
- Use C++.
  - It appears to dominate the field as far as the tools of choice for discriminating hackers are concerned.
  - Used Haskell this year.

# How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- ~~If you want to do well – get on a strong team.~~
- ~~Don't forget to rest, eat and sleep.~~
- ~~Random tweaks don't work. Think instead.~~
- ~~Read the spec carefully.~~
- ~~Use C++.~~

# How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- ~~If you want to do well – get on a strong team.~~
- ~~Don't forget to rest, eat and sleep.~~
- ~~Random tweaks don't work. Think instead.~~
- ~~Read the spec carefully.~~
- ~~Use C++.~~
- All in all, I have no idea how does this work.

## Lots of teams from Eastern Europe and Russia for some reason?

- In 2012 the organizers expressed puzzlement with a somewhat skewed geographical distribution of teams.



## Lots of teams from Eastern Europe and Russia for some reason?

- In 2012 the organizers expressed puzzlement with a somewhat skewed geographical distribution of teams.
- Dmitry Astapov's ICFPC reports  
([http://users.livejournal.com/\\_adept\\_/tag/icfpc](http://users.livejournal.com/_adept_/tag/icfpc)) – in Russian

# Lots of teams from Eastern Europe and Russia for some reason?

- In 2012 the organizers expressed puzzlement with a somewhat skewed geographical distribution of teams.
- Dmitry Astapov's ICFPC reports  
([http://users.livejournal.com/\\_adept\\_/tag/icfpc](http://users.livejournal.com/_adept_/tag/icfpc)) – in Russian
- Better reading than most fiction.

## Lots of teams from Eastern Europe and Russia for some reason?

- In 2012 the organizers expressed puzzlement with a somewhat skewed geographical distribution of teams.
- Dmitry Astapov's ICFPC reports ([http://users.livejournal.com/\\_adept\\_/tag/icfpc](http://users.livejournal.com/_adept_/tag/icfpc)) – in Russian
- Better reading than most fiction.
- The 2006 one stands out in particular (and so did the contest itself).

# Lots of teams from Eastern Europe and Russia for some reason?

- In 2012 the organizers expressed puzzlement with a somewhat skewed geographical distribution of teams.
- Dmitry Astapov's ICFPC reports ([http://users.livejournal.com/\\_adept\\_/tag/icfpc](http://users.livejournal.com/_adept_/tag/icfpc)) – in Russian
- Better reading than most fiction.
- The 2006 one stands out in particular (and so did the contest itself).
- Many folks who read those reports were instantly hooked and wanted to give it a shot – myself included.

# Lots of teams from Eastern Europe and Russia for some reason?

- In 2012 the organizers expressed puzzlement with a somewhat skewed geographical distribution of teams.
- Dmitry Astapov's ICFPC reports ([http://users.livejournal.com/\\_adept\\_/tag/icfpc](http://users.livejournal.com/_adept_/tag/icfpc)) – in Russian
- Better reading than most fiction.
- The 2006 one stands out in particular (and so did the contest itself).
- Many folks who read those reports were instantly hooked and wanted to give it a shot – myself included.
- Now you know whom to blame.

# Compilers Are Scary Stuff

- Considered to be a dark art among the uninitiated.

# Compilers Are Scary Stuff

- Considered to be a dark art among the uninitiated.
- Dr. Alex Aiken and his team are running an open Compilers class on Coursera – <https://www.coursera.org/course/compilers>

# Compilers Are Scary Stuff

- Considered to be a dark art among the uninitiated.
- Dr. Alex Aiken and his team are running an open Compilers class on Coursera – <https://www.coursera.org/course/compilers>
- It's a good one. Get through it and you'll never be scared of writing a compiler again.



# Compilers Are Scary Stuff

- Considered to be a dark art among the uninitiated.
- Dr. Alex Aiken and his team are running an open Compilers class on Coursera – <https://www.coursera.org/course/compilers>
- It's a good one. Get through it and you'll never be scared of writing a compiler again.
- **Correction:** writing a *toy* compiler.

# Compilers Are Scary Stuff

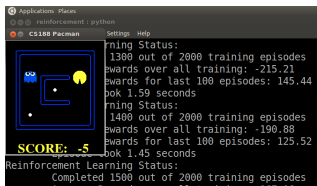
- Considered to be a dark art among the uninitiated.
- Dr. Alex Aiken and his team are running an open Compilers class on Coursera – <https://www.coursera.org/course/compilers>
- It's a good one. Get through it and you'll never be scared of writing a compiler again.
- **Correction:** writing a *toy* compiler.
- Thankfully, the GHC and GCC in this contest were not the Real Thing.

# AI Is A Bit Scary Too

- Dr. Dan Klein and others developed an Artificial Intelligence class on edX – <https://www.edx.org/course/uc-berkeleyx/uc-berkeleyx-cs188-1x-artificial-579>

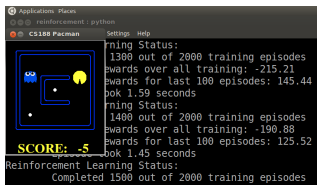
# AI Is A Bit Scary Too

- Dr. Dan Klein and others developed an Artificial Intelligence class on edX – <https://www.edx.org/course/uc-berkeleyx/uc-berkeleyx-cs188-1x-artificial-579>
- Curiously, it uses Pac-Man in examples and assignments a lot.



# AI Is A Bit Scary Too

- Dr. Dan Klein and others developed an Artificial Intelligence class on edX – <https://www.edx.org/course/uc-berkeleyx/uc-berkeleyx-cs188-1x-artificial-579>
- Curiously, it uses Pac-Man in examples and assignments a lot.



- I'm sure that's sheer coincidence and had *nothing* to do with my performance here.

**This was even more awesome than ICFP contests usually are.  
Heartfelt thanks to the organizers.  
And thank you for listening.**

**And remember – for the next year, Haskell is the programming tool of choice for discriminating hackers.**