ICFP Programming Contest 2010

International Cars and Fuels Production

Bertram Felgenhauer, University of Innsbruck, Austria
Johannes Waldmann, HTWK Leipzig, Germany

June 18–21, 2010

About the ICFP Programming Contest

- programming, problem solving, fun
- annual contest, since 1998
- sponsored by ICFP conference/ACM
- 2010 contest hosted by HTWK Leipzig, Germany
- contest format
  - 72 hours (June 18, 12:00 – June 21, 12:00 GMT)
  - participation online, international
  - teams allowed
  - no fixed programming language
  - lightning division (first 24 hours)

Contest Task

storyline:

- market for
  - cars (= problem instance) (public)
  - fuels (= problem solution) (private)

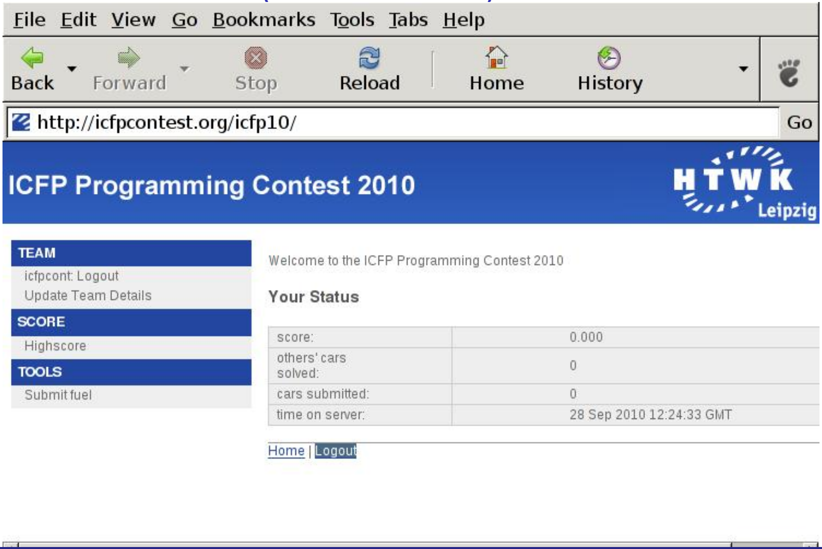
Contest Task

storyline:

- market for
  - cars (= problem instance) (public)
  - fuels (= problem solution) (private)
- earn money by
  - (efficiently) solving instances,
  - or creating instances (with solution, which is hard to find)

  income tax (devaluates earnings by 1/2 per day)

Contest Web GUI (Team Status)



Contest Task

storyline:

- market for
  - cars (= problem instance) (public)
  - fuels (= problem solution) (private)
- earn money by
  - (efficiently) solving instances,
  - or creating instances (with solution, which is hard to find)

  income tax (devaluates earnings by 1/2 per day)

obfuscation:

Contest Task

storyline:

- market for
  - cars (= problem instance) (public)
  - fuels (= problem solution) (private)
- earn money by
  - (efficiently) solving instances,
  - or creating instances (with solution, which is hard to find)

  income tax (devaluates earnings by 1/2 per day)

obfuscation:

- ternary stream encoding of structured data
  (= format used for published instances)

Contest Task

storyline:

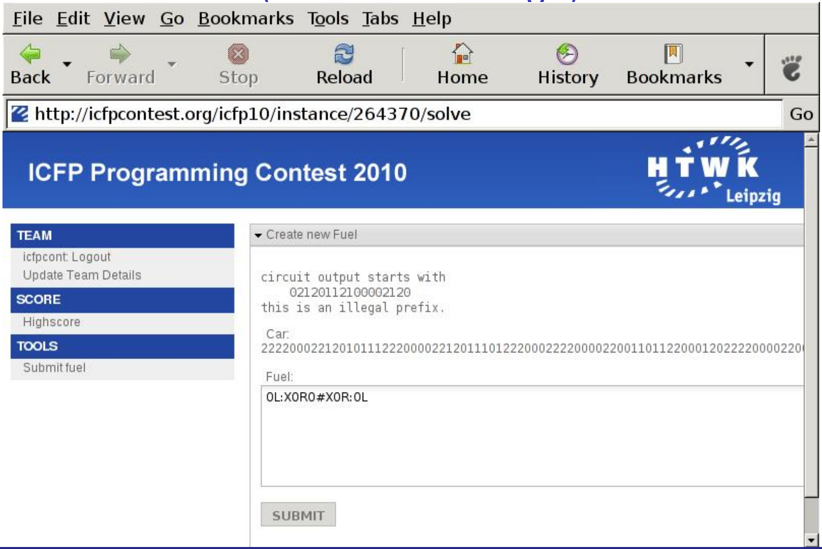
- market for
  - cars (= problem instance) (public)
  - fuels (= problem solution) (private)
- earn money by
  - (efficiently) solving instances,
  - or creating instances (with solution, which is hard to find)

  income tax (devaluates earnings by 1/2 per day)

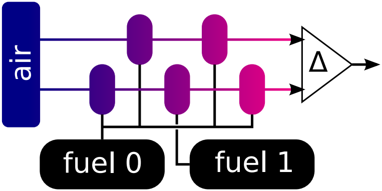
obfuscation:

- ternary stream encoding of structured data
  (= format used for published instances)
- ternary circuits (with state) to produce streams
  (= format used for submitted solutions)

Contest Web GUI (Submission Page)

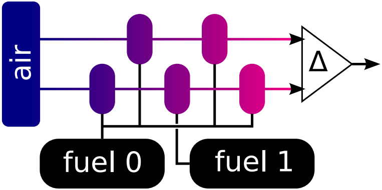


The Difference Engine that Moves Cars



- air: contains several ingredients:
  vector over $\mathbb{N}$, first component positive
- fuel component:
  (linearly) transforms incoming air in reaction chamber:
  square matrix over $\mathbb{N}$, top left entry positive
- difference engine compares upper and lower pipe's outputs: $(\geq)$ everywhere, $(>)$ in first component.

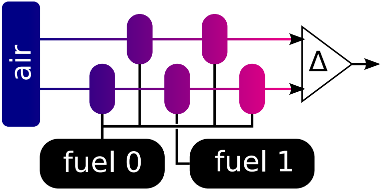
Example Car and Fuel



In other words,

$\mathbb{M} :=$ *square matrices over* $\mathbb{N}$ *with top left entry* ≥ 1
Find $A, B \in \mathbb{M}$ *with* $AA - ABA \in \mathbb{M}$.

Example Car and Fuel



In other words,

$\mathbb{M} :=$ *square matrices over* $\mathbb{N}$ *with top left entry* ≥ 1
Find $A, B \in \mathbb{M}$ *with* $AA - ABA \in \mathbb{M}$.

$$[\![\text{fuel 0}]\!] = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}, \ [\![\text{fuel 1}]\!] = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

$$[\![\text{output}]\!] = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix} - \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

Ternary Encoding: Definition

22102200002201011220101002200112201110220010

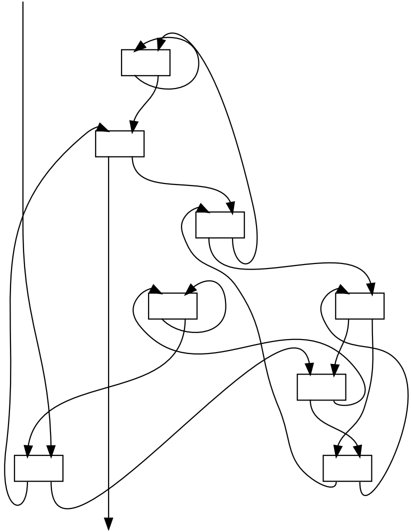
- types: car = $[([\mathbb{N}], \mathbb{N}, [\mathbb{N}])]$, fuel = $[[[\mathbb{N}]]]$
- tuples: $[\![(a, b)]\!] = [\![a]\!][\![b]\!]$
- lists:
  - $[\![\text{nil}]\!] = 0$, $[\![x : \text{nil}]\!] = 1[\![x]\!]$
  - $[\![x]\!] = 22[\![\text{len}(x) - 2]\!][\![x_1]\!] \dots [\![x_{\text{len}(x)}]\!]$
- natural numbers: $[\![n]\!] = [\![raw(n)]\!]$
  - $raw(0) = \text{nil}$
  - $raw(n) = (n-1) \bmod 3 : raw((n-1) \operatorname{div} 3)$
  - 0, 10, 11, 12, 2 0 00. . . 2 0 22, 2 10 000. . . 2 10 222, . . .
- contestants had to reverse engineer the encoding (from parser's error messages) (the 22 gives some redundancy)

Ternary Encoding: Implementation

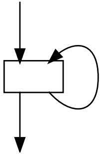
- use Parsec (of course)
- use Printer/Parser pairs

```
data PrinterParser t a =
     PrinterParser
        { printer :: a -> Maybe [t]
        , parser  :: Text.Parsec.Parsec [t] () a
        }
or :: PrinterParser t a
   -> PrinterParser t b
   -> PrinterParser t (Either a b)
...
```

obfuscation: circuits



obfuscation: circuits

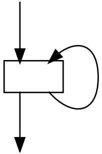


```
code:    0L:X0R0#X0R:0L

input:   01202101210201202...
output:  02120112100002120...
```

- one external input, one external output
- each gate has left and right inputs and outputs
- every output is connected to exactly one input
- backwards wires are delayed
- contestants had to deduce the circuit synatx (from example and error messages) and gate semantics (from our simulator's output)
- then build their own simulator, and circuit compiler.
- semantics:

obfuscation: circuits



```
code:    0L:X0R0#X0R:0L

input:   0120210121020120 2...
output:  0212011210000212 0...
```

- one external input, one external output
- each gate has left and right inputs and outputs
- every output is connected to exactly one input
- backwards wires are delayed
- contestants had to deduce the circuit synatx (from example and error messages) and gate semantics (from our simulator's output)
- then build their own simulator, and circuit compiler.
- semantics: left: $(l - r) \bmod 3$, right: $(l \cdot r - 1) \bmod 3$

Participation

- 871 teams
- 214 teams figured out the circuit
- 146 teams submitted valid fuels
- 3,746 submitted cars
- 257,901 fuels (i.e. correct solutions)
- 350,344 bytes: max fuel (circuit description)
- 22,889 bytes: max car (ternary string)

Statistical Data

Country	Language	(average) Age

Statistical Data

Country Language (average) Age

29	USA
28	Japan
25	Russia
12	Germany
11	Ukraine
8	France
7	UK
5	Hungary
4	Australia
4	Canada

Statistical Data

Country

29	USA
28	Japan
25	Russia
12	Germany
11	Ukraine
8	France
7	UK
5	Hungary
4	Australia
4	Canada

Language

29	Haskell
17	C++
16	Python
12	Java
7	OCaml
6	Ruby
5	C#
3	Common Lisp

(average) Age

Statistical Data

Country

29	USA
28	Japan
25	Russia
12	Germany
11	Ukraine
8	France
7	UK
5	Hungary
4	Australia
4	Canada

Language

29	Haskell
17	C++
16	Python
12	Java
7	OCaml
6	Ruby
5	C#
3	Common Lisp

(average) Age

- 81734636191301391914
- 24.333333333333
- grad student
- DRINKING AGE
- grey-beard
- 22 sucks!
- 54.043189
- below 30
- unknown
- 22,23
- 0x20
- 31.3
- 15.5

The 2010 Contest Team

at HTWK Leipzig:

- Web server programming and maintenance:
  Daniel Borkmann, Tobias Kalbitz, Christopher Schädlich, Michael Schmeißer
- Web design, Brute force solver: Johannes Erber
- Log file evaluation: Christian Reichmann
- Contest task design, semantics server programming:
  Bertram Felgenhauer, Johannes Waldmann

external:

- testers:
  Alexander Kiel (Univ. Leipzig), Georg Martius (Univ. Göttingen), Henning Thielemann (Univ. Halle)
- advisor:
  Robby Findler (Northwestern Univ.)

Hard- and Software

specs:

- Cluster of 5 $\times$ Intel(R) Xeon(R) CPU X5365 @ 3.00GHz
- Debian GNU/Linux OS
- Web server (1 node):
  apache2, tomcat, Spring(Roo)/Java, postgresql
- Semantics server (4 nodes):
  haskell (xmlrpc, parsec, autolib)

load:

- 115,760 submissions per hour: peak server load
- 20 GB: total incoming traffic
- 768,049,967 bytes: size of gzipped contest database

References/Reverences

- input stream `01202101210201202...`

References/Reverences

- input stream `01202101210201202...` is the *ternary Morse-Thue sequence*, a squarefree D0L sequence

```
let f 0 = [0,1,2] ; f 1 = [0,2] ; f 2 = [1]
    rest = 1 : 2 : ( rest >>= f )
in  0 : rest
```

References/Reverences

- input stream `0120210121020120 2...` is the *ternary Morse-Thue sequence*, a squarefree D0L sequence

```
let f 0 = [0,1,2] ; f 1 = [0,2] ; f 2 = [1]
    rest = 1 : 2 : ( rest >>= f )
in  0 : rest
```

- ```
  module ICFP.Config where
  required_prefix = [1,1,0,2,1,2,1,0,1,1,2,1,0,1,2,2,1
  ```
  ```

References/Reverences

- input stream `0120210121020120 2...` is the *ternary Morse-Thue sequence*, a squarefree D0L sequence

```
let f 0 = [0,1,2] ; f 1 = [0,2] ; f 2 = [1]
    rest = 1 : 2 : ( rest >>= f )
in  0 : rest
```

- `module ICFP.Config where`
  `required_prefix = [1,1,0,2,1,2,1,0,1,1,2,1,0,1,2,2,1`
  is `"ICFP"` in morse code

References/Reverences

- input stream `0120210121020120 2...` is the *ternary Morse-Thue sequence*, a squarefree D0L sequence

```
let f 0 = [0,1,2] ; f 1 = [0,2] ; f 2 = [1]
    rest = 1 : 2 : ( rest >>= f )
in  0 : rest
```

- `module ICFP.Config where`
  `required_prefix = [1,1,0,2,1,2,1,0,1,1,2,1,0,1,2,2,1`
  is `"ICFP"` in morse code
- *cars*: cf. racing track of 2003 contest (Chalmers)
- *circuits*: cf. 2D (ASCII) programming in 2006 (CMU)
- *prefix*: cf. 2007 contest (Utrecht)

Comments from Participants

- I really enjoyed this problem set. Our team had a ton of fun.

Comments from Participants

- I really enjoyed this problem set. Our team had a ton of fun.
- Thanks for the fun competition! You did a great job with the problem this year. It was fun to unravel layer after layer of it!

Comments from Participants

- I really enjoyed this problem set. Our team had a ton of fun.
- Thanks for the fun competition! You did a great job with the problem this year. It was fun to unravel layer after layer of it!
- I enjoyed the contest very much; while the server problems were frustrating and discouraging, we think the task was very well crafted and interesting.

Comments from Participants

- I really enjoyed this problem set. Our team had a ton of fun.
- Thanks for the fun competition! You did a great job with the problem this year. It was fun to unravel layer after layer of it!
- I enjoyed the contest very much; while the server problems were frustrating and discouraging, we think the task was very well crafted and interesting.
- We had a lot of fun. The subject was great, with a good balance between mathematics and programming. We also liked the reverse engineering approach.

Comments from Participants

- I really enjoyed this problem set. Our team had a ton of fun.
- Thanks for the fun competition! You did a great job with the problem this year. It was fun to unravel layer after layer of it!
- I enjoyed the contest very much; while the server problems were frustrating and discouraging, we think the task was very well crafted and interesting.
- We had a lot of fun. The subject was great, with a good balance between mathematics and programming. We also liked the reverse engineering approach.
- We have been participating to the ICFP programming contest for years and your subject is among the best ones.

Comments from Participants

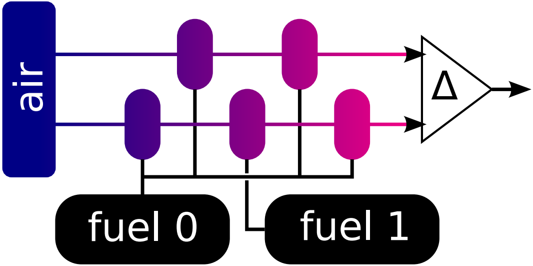
- I really enjoyed this problem set. Our team had a ton of fun.
- Thanks for the fun competition! You did a great job with the problem this year. It was fun to unravel layer after layer of it!
- I enjoyed the contest very much; while the server problems were frustrating and discouraging, we think the task was very well crafted and interesting.
- We had a lot of fun. The subject was great, with a good balance between mathematics and programming. We also liked the reverse engineering approach.
- We have been participating to the ICFP programming contest for years and your subject is among the best ones.
- The problems with the server were frustrating (for everybody, including organizers, I'm sure).

Comments from Participants

- I really enjoyed this problem set. Our team had a ton of fun.
- Thanks for the fun competition! You did a great job with the problem this year. It was fun to unravel layer after layer of it!
- I enjoyed the contest very much; while the server problems were frustrating and discouraging, we think the task was very well crafted and interesting.
- We had a lot of fun. The subject was great, with a good balance between mathematics and programming. We also liked the reverse engineering approach.
- We have been participating to the ICFP programming contest for years and your subject is among the best ones.
- The problems with the server were frustrating (for everybody, including organizers, I'm sure).
- Your servers are failing so hard, you ruined the contest!

Background Story: Cars and Rewriting

- chambers of engine = rules



- upper, lower pipe = lhs, rhs of rewriting rule: $00 \rightarrow 010$
- *fuel* = matrix interpretation, a method of proving termination

Termination of Rewriting

...is undecidable, but important for software verification
automated termination provers:

- Termination Competition (termcomp) (yearly, since 2004)
- Termination Problem Data Base (TPDB)

methods for automated termination analysis:

- syntactic, e.g.,
  - recursive path orders (Dershowitz, 1982)
- semantic (interpretation), e.g.,
  - polynomial functions over $\mathbb{N}$ (Lankford, 1979)
  - linear functions ($=$ matrices) over vectors over $\mathbb{N}$ (Hofbauer, Waldmann, 2006)

How to Find Matrix Interpretations

$\mathbb{M} :=$ square matrices over $\mathbb{N}$ with top left entry ≥ 1

- Find $A, B \in \mathbb{M}$ with $AA - ABA \in \mathbb{M}$.
- Find $A, B \in \mathbb{M}$ with $A^2B^2 - B^3A^3 \in \mathbb{M}$.
- ...

Participants used these methods:

- brute force (complete or random enumeration)

How to Find Matrix Interpretations

$\mathbb{M} :=$ square matrices over $\mathbb{N}$ with top left entry ≥ 1

- Find $A, B \in \mathbb{M}$ with $AA - ABA \in \mathbb{M}$.
- Find $A, B \in \mathbb{M}$ with $A^2B^2 - B^3A^3 \in \mathbb{M}$.
- ...

Participants used these methods:

- brute force (complete or random enumeration)
- built-in solvers of computer algebra systems

How to Find Matrix Interpretations

$\mathbb{M} :=$ square matrices over $\mathbb{N}$ with top left entry ≥ 1

- Find $A, B \in \mathbb{M}$ with $AA - ABA \in \mathbb{M}$.
- Find $A, B \in \mathbb{M}$ with $A^2B^2 - B^3A^3 \in \mathbb{M}$.
- ...

Participants used these methods:

- brute force (complete or random enumeration)
- built-in solvers of computer algebra systems
- linear programming for 1-dimensional matrices (after taking logarithms)

How to Find Matrix Interpretations

$\mathbb{M}$:= square matrices over $\mathbb{N}$ with top left entry ≥ 1

- Find $A, B \in \mathbb{M}$ with $AA - ABA \in \mathbb{M}$.
- Find $A, B \in \mathbb{M}$ with $A^2B^2 - B^3A^3 \in \mathbb{M}$.
- ...

Participants used these methods:

- brute force (complete or random enumeration)
- built-in solvers of computer algebra systems
- linear programming for 1-dimensional matrices (after taking logarithms)
- simulated annealing for higher dimensions
- similar randomized hill climbing approaches

How to Find Matrix Interpretations

$\mathbb{M} :=$ square matrices over $\mathbb{N}$ with top left entry ≥ 1

- Find $A, B \in \mathbb{M}$ with $AA - ABA \in \mathbb{M}$.
- Find $A, B \in \mathbb{M}$ with $A^2B^2 - B^3A^3 \in \mathbb{M}$.
- ...

Participants used these methods:

- brute force (complete or random enumeration)
- built-in solvers of computer algebra systems
- linear programming for 1-dimensional matrices (after taking logarithms)
- simulated annealing for higher dimensions
- similar randomized hill climbing approaches
- (SMT solvers? SAT encoding?)

How to Produce Hard Termination Problems

…that still can be solved by matrix interpretation

How to Produce Hard Termination Problems

...that still can be solved by matrix interpretation

- "trivial" problems like $1 \succ 22, 2 \succ 33, \ldots$
  (needed to post these early, or often)

How to Produce Hard Termination Problems

…that still can be solved by matrix interpretation

- "trivial" problems like $1 \succ 22, 2 \succ 33, \dots$
  (needed to post these early, or often)
- systematic constructions
  $0121 \succ 1211012012$,
  $0121 \succ 1211012012012$, …
  (can all be solved by the same interpretation)

How to Produce Hard Termination Problems

…that still can be solved by matrix interpretation

- "trivial" problems like $1 \succ 22, 2 \succ 33, \ldots$
  (needed to post these early, or often)
- systematic constructions
  $0121 \succ 1211012012$,
  $0121 \succ 1211012012012$, …
  (can all be solved by the same interpretation)
- encoding of diophantine equations
  (but: no easy way to enforce the intended shape/semantics)

How to Produce Hard Termination Problems

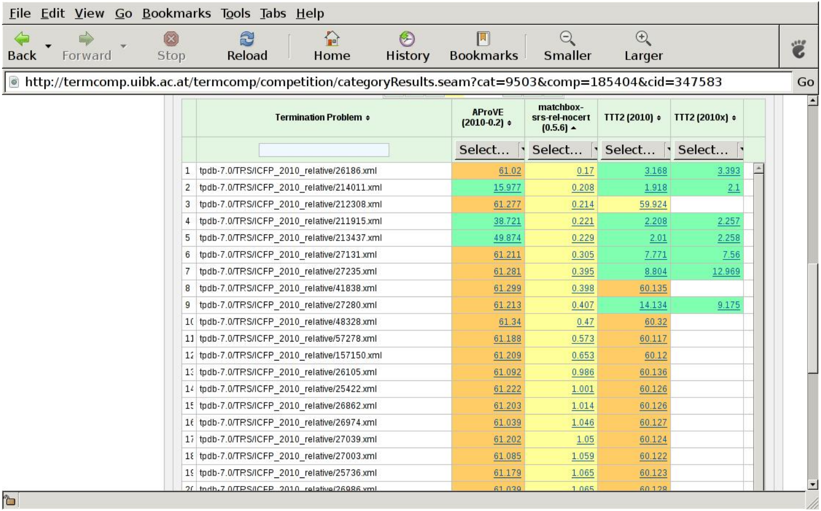
...that still can be solved by matrix interpretation

- "trivial" problems like $1 \succ 22, 2 \succ 33, \ldots$
  (needed to post these early, or often)
- systematic constructions
  $0121 \succ 1211012012$,
  $0121 \succ 1211012012012, \ldots$
  (can all be solved by the same interpretation)
- encoding of diophantine equations
  (but: no easy way to enforce the intended shape/semantics)
- take set of random sparse matrices (fuel), then generate matching cars. prefer length increasing rules.
  $5305453 \succeq 5510450343, 5412501 \succ 3343403001$

After the Contest

- selection of ICFP contest problems was submitted to TPDB,
- were used in Termination Competition 2010 (July): termination provers performed badly (only very few were solved)

ICFP contest problems in Termcomp



	Termination Problem	AProVE (2010-0.2)	matchbox-srs-rel-nocert (0.5.6)	TTT2 (2010)	TTT2 (2010x)
		Select...	Select...	Select...	Select...
1	tpdb-7.0/TRS/ICFP_2010_relative/26186.xml	61.02	0.17	3.168	3.393
2	tpdb-7.0/TRS/ICFP_2010_relative/214011.xml	15.977	0.208	1.918	2.1
3	tpdb-7.0/TRS/ICFP_2010_relative/212308.xml	61.277	0.214	59.924	
4	tpdb-7.0/TRS/ICFP_2010_relative/211915.xml	38.721	0.221	2.208	2.257
5	tpdb-7.0/TRS/ICFP_2010_relative/213437.xml	49.874	0.229	2.01	2.258
6	tpdb-7.0/TRS/ICFP_2010_relative/27131.xml	61.211	0.305	7.771	7.56
7	tpdb-7.0/TRS/ICFP_2010_relative/27235.xml	61.281	0.395	8.804	12.969
8	tpdb-7.0/TRS/ICFP_2010_relative/41838.xml	61.299	0.398	60.135	
9	tpdb-7.0/TRS/ICFP_2010_relative/27280.xml	61.213	0.407	14.134	9.175
10	tpdb-7.0/TRS/ICFP_2010_relative/48328.xml	61.34	0.47	60.32	
11	tpdb-7.0/TRS/ICFP_2010_relative/57278.xml	61.188	0.573	60.117	
12	tpdb-7.0/TRS/ICFP_2010_relative/157150.xml	61.209	0.653	60.12	
13	tpdb-7.0/TRS/ICFP_2010_relative/26105.xml	61.092	0.986	60.136	
14	tpdb-7.0/TRS/ICFP_2010_relative/25422.xml	61.222	1.001	60.126	
15	tpdb-7.0/TRS/ICFP_2010_relative/26862.xml	61.203	1.014	60.126	
16	tpdb-7.0/TRS/ICFP_2010_relative/26974.xml	61.039	1.046	60.127	
17	tpdb-7.0/TRS/ICFP_2010_relative/27039.xml	61.202	1.05	60.124	
18	tpdb-7.0/TRS/ICFP_2010_relative/27003.xml	61.085	1.059	60.122	
19	tpdb-7.0/TRS/ICFP_2010_relative/25736.xml	61.179	1.065	60.123	
20	tpdb-7.0/TRS/ICFP_2010_relative/26986.xml	61.039	1.065	60.128	

After the Contest

- selection of ICFP contest problems was submitted to TPDB,
- were used in Termination Competition 2010 (July): termination provers performed badly (only very few were solved)
- contest participants should consider entering their matrix solver into next termination competition (use one of the open-sourced termination tools and plug in your solver)

`http://termination-portal.org/wiki/Termination_Competition/`

After the Contest

solve this puzzle:

- $\mathbb{M} :=$ square matrices over $\mathbb{N}$ with top left entry ≥ 1
  Find $A, B \in \mathbb{M}$ with $A^2 B^2 - B^3 A^3 \in \mathbb{M}$.

After the Contest

solve this puzzle:

- $\mathbb{M} :=$ square matrices over $\mathbb{N}$ with top left entry ≥ 1
  Find $A, B \in \mathbb{M}$ with $A^2B^2 - B^3A^3 \in \mathbb{M}$.

prove (or disprove and repair) this theorem:

- If a string rewriting system admits an $\mathbb{M}$-interpretation, then its *derivational complexity* (max. derivation length, as function of start term size) is linear.

After the Contest

solve this puzzle:

- $\mathbb{M} :=$ square matrices over $\mathbb{N}$ with top left entry ≥ 1
  Find $A, B \in \mathbb{M}$ with $A^2B^2 - B^3A^3 \in \mathbb{M}$.

prove (or disprove and repair) this theorem:

- If a string rewriting system admits an $\mathbb{M}$-interpretation, then its *derivational complexity* (max. derivation length, as function of start term size) is linear.

and submit paper to

- Intl. Workshop on Termination
  (next: February 2012, near Innsbruck, Austria)
  `http://termination-portal.org/wiki/WST/`
- Conf. Rewriting Techniques and Applications (RTA)
  (next: July 2011, Novi Sad; May 2012, Nagoya)
  `http://rewriting.loria.fr/rta/`

and now . . .

and now . . .

. . . the ICFP 2010 programming contest winners

Judges' Prize

for their very efficient circuit encoding (one gate per trit)

Cult of the Bound Variable

. . . are an extremely cool bunch of hackers.

Languages: SML, C++

actually also Python, Mathematica, AMPL, Perl, bash, and PHP.

. . . figured out our final circuit encoding in a fit of brilliance at 2am Saturday morning (and finished implementing it in C++ before anyone who knew SML woke up, which ended up being fine)

Winner of the Lightning Division

best score after 24 hours
produced very hard problem instances

Carl Witty (team Witrala)

Language: Sage (`http://www.sagemath.org/`),
a computer algebra system that runs under Python.

Sage is very suitable for rapid prototyping. . .
and a fine tool for many applications
(Carl got second best score after 72 hours)

Winner of the Main Division

best score after 72 hours

Pure Pure Code ++

Languages: C++, Haskell, Python

. . . are the programming languages of choice for discriminating hackers.

The Future

- the ICFP 2011 programming contest will be run by *Eijiro Sumii* at *Tohoku University*.
- (Tell your students to)
  Take part in the ICFP programming contest!
- all information via
  - `http://icfpcontest.org/`
  - `http://www.icfpconference.org/`