Unleashing the verification genie in the cloud

Nikolaj Bjørner & Leonardo de Moura
Microsoft Research

NSF Usable Verification Workshop Nov 15-16 2010

FSE & RiSE

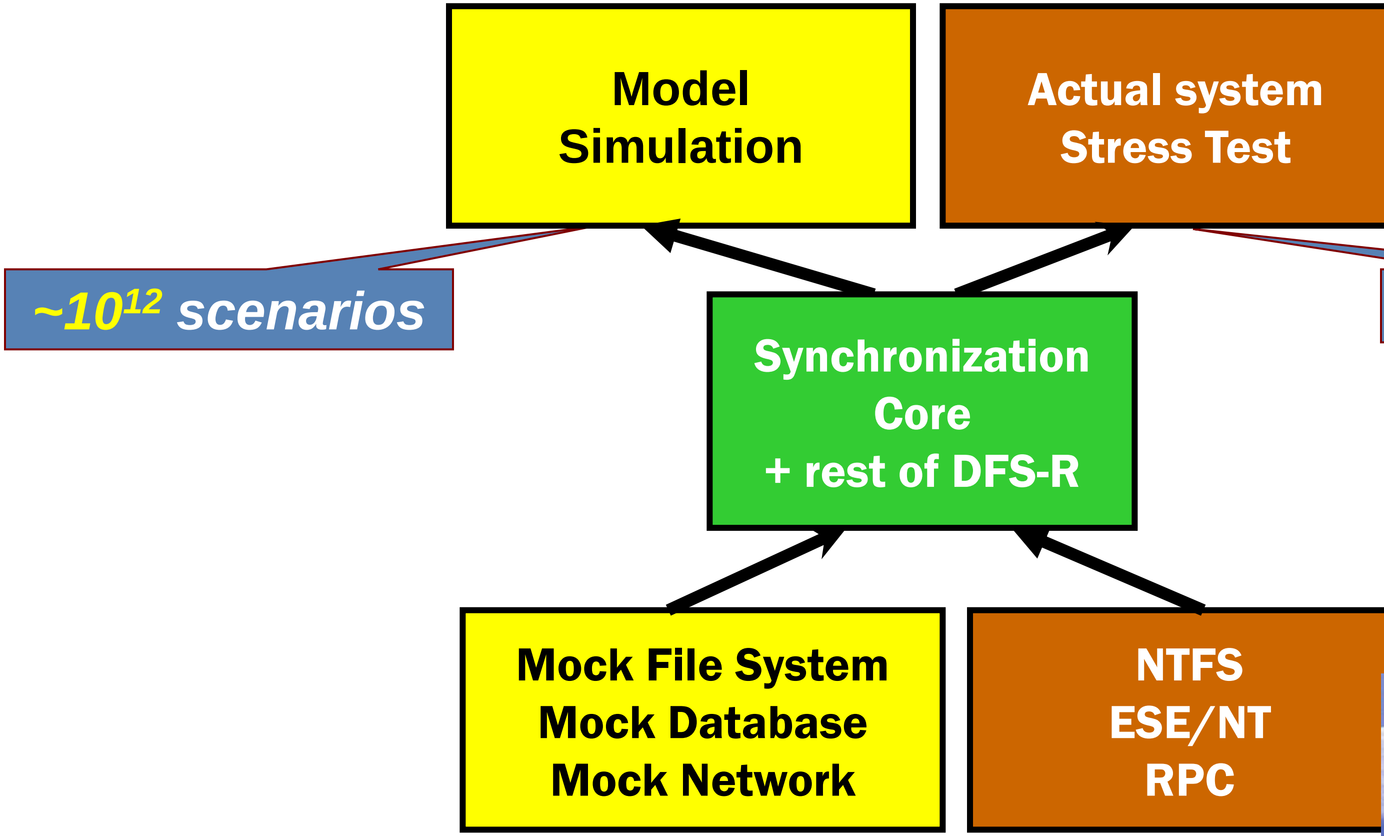
DFS-R: “hands on” usability

CAML validator example run

```
C:\>.\frsmodel -bug -b2 6
…
Count=57000
Child with Id: f_6 does not exist in DB
Invariant violated db_is_consistent_with_fs

         Machine 1       Machine 2
----------------------------------------
1        create q
2        create q\b
3                        create q
4        flush_journal
5                        sync from m_1
6        sync from m_2
```

DFS-R: “hands on” usability



Model Simulation

Actual system Stress Test

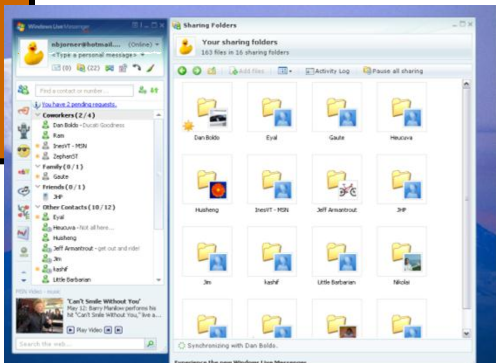
~10^{12} scenarios

~10^{5} scenarios

Synchronization Core + rest of DFS-R

Mock File System
Mock Database
Mock Network

NTFS
ESE/NT
RPC


Windows Live Messenger
nbjorner@hotmail.... (Online)
<Type a personal message>
(0) (22)
Find a contact or number...
You have 2 pending requests.
Coworkers (2 / 4)
Dan Boldo - Ducati Goodness
Ram
InesVT - MSN
ZephanST
Family (0 / 1)
Gaute
Friends (0 / 1)
JHP
Other Contacts (10 / 12)
Eyal
Heucuva - Not all here...
Huisheng
Jeff Armantrout - get out and ride!
Jim
kashif
Little Barbarian
MSN Video - music
'Can't Smile Without You'
May 12: Barry Manilow performs his hit "Can't Smile Without You," live a...
Play Video
Search the web...
Sharing Folders
Your sharing folders
163 files in 16 sharing folders
Add files
Activity Log
Pause all sharing
Dan Boldo
Eyal
Gaute
Heucuva
Huisheng
InesVT - MSN
Jeff Armantrout
JHP
Jim
kashif
Little Barbarian
Nikolai
Synchronizing with Dan Boldo.
Experience the new Windows Live Messenger

State space exploration on production code:
200 machines x 2 weeks = ½ trillion scenarios

Session Focus

Improved automation:

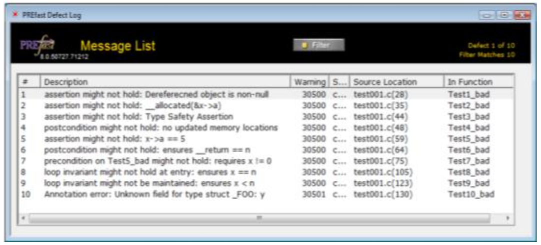
- Usable automatic answers
- Efficiency and expressivity in Z3

Delivering and combining inference capabilities:

- SMT-LIB2@ http://rise4fun.com/z3, LINQ, Quotations, Boogie
  … and other ways of lowering the barrier of entry for using Z3
- Z3 user-based theory solvers

Some Microsoft Engines using Z3

- **SDV:** The Static Driver Verifier
- **PREfix:** The Static Analysis Engine for C/C++.
- **Pex:** Program EXploration for .NET.
- **SAGE:** Scalable Automated Guided Execution
- **Spec#:** C# + contracts
- **VCC:** Verifying C Compiler for the Viridian Hyper-Visor
- **HAVOC:** Heap-Aware Verification of C-code.
- **SpecExplorer:** Model-based testing of protocol specs.
- **Yogi:** Dynamic symbolic execution + abstraction.
- **FORMULA:** Model-based Design
- **F7:** Refinement types for security protocols
- **Rex:** Regular Expressions and formal languages
- **VS3:** Abstract interpretation and Synthesis
- **VERVE:** Verified operating system
- **FINE:** Proof carrying certified code

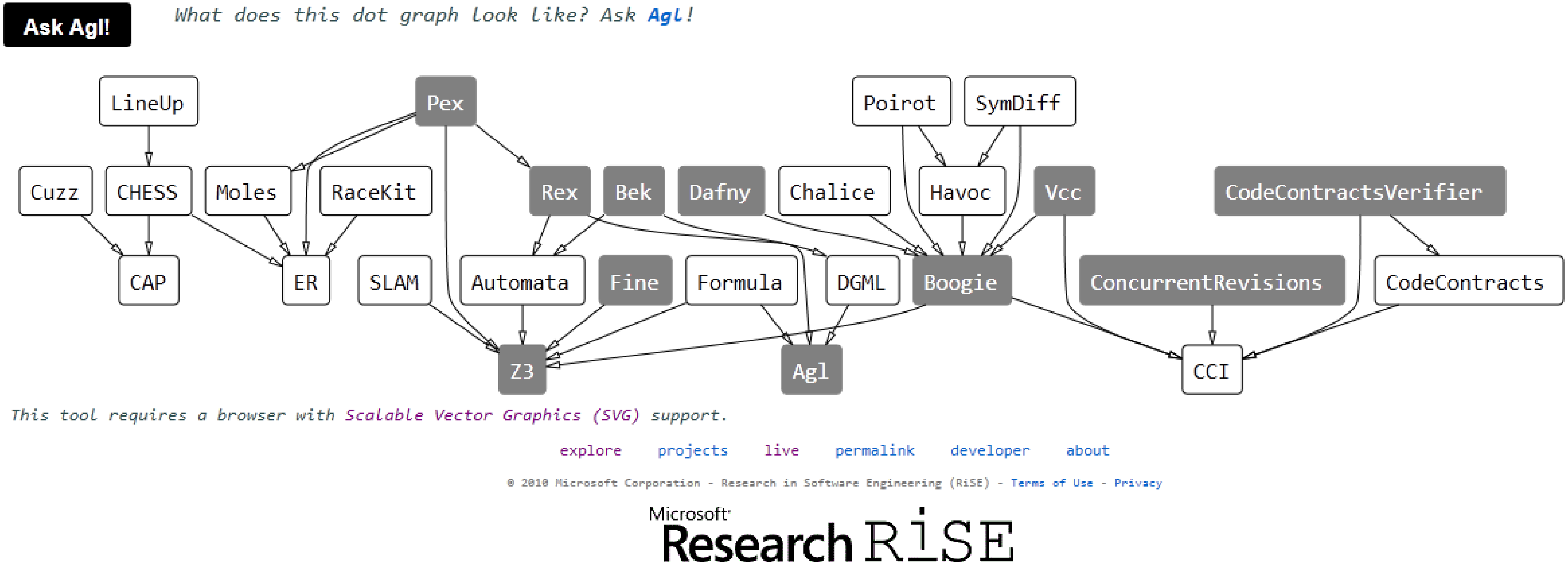


Message List

Defect 1 of 10
Filter Matches 10

#	Description	Warning	S...	Source Location	In Function
1	assertion might not hold: Dereferecned object is non-null	30500	c...	test001.c(28)	Test1_bad
2	assertion might not hold: __allocated(&x->a)	30500	c...	test001.c(35)	Test2_bad
3	assertion might not hold: Type Safety Assertion	30500	c...	test001.c(44)	Test3_bad
4	postcondition might not hold: no updated memory locations	30500	c...	test001.c(48)	Test4_bad
5	assertion might not hold: x->a == 5	30500	c...	test001.c(59)	Test5_bad
6	postcondition might not hold: ensures __return == n	30500	c...	test001.c(64)	Test6_bad
7	precondition on Test5_bad might not hold: requires x != 0	30500	c...	test001.c(75)	Test7_bad
8	loop invariant might not hold at entry: ensures x == n	30500	c...	test001.c(105)	Test8_bad
9	loop invariant might not be maintained: ensures x < n	30500	c...	test001.c(123)	Test9_bad
10	Annotation error: Unknown field for type struct _FOO: y	30501	c...	test001.c(130)	Test10_bad

RiSE tool chain

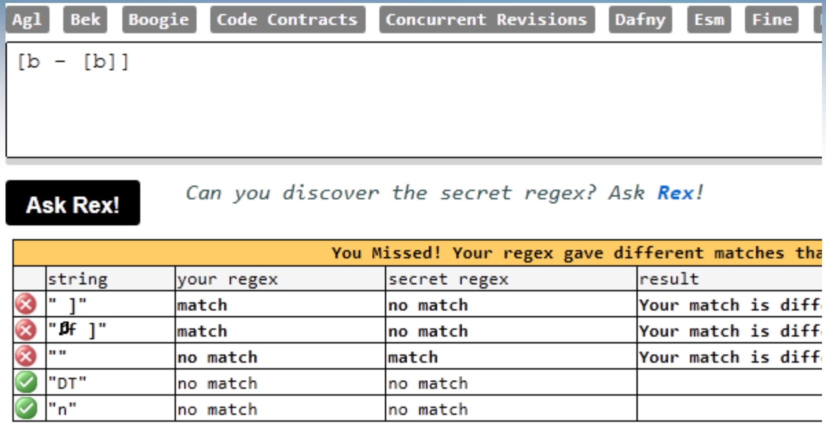


*What does this dot graph look like? Ask **Agl**!*

This tool requires a browser with Scalable Vector Graphics (SVG) support.

© 2010 Microsoft Corporation - Research in Software Engineering (RiSE) - Terms of Use - Privacy

Usable Automatic Answers

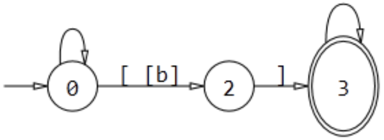


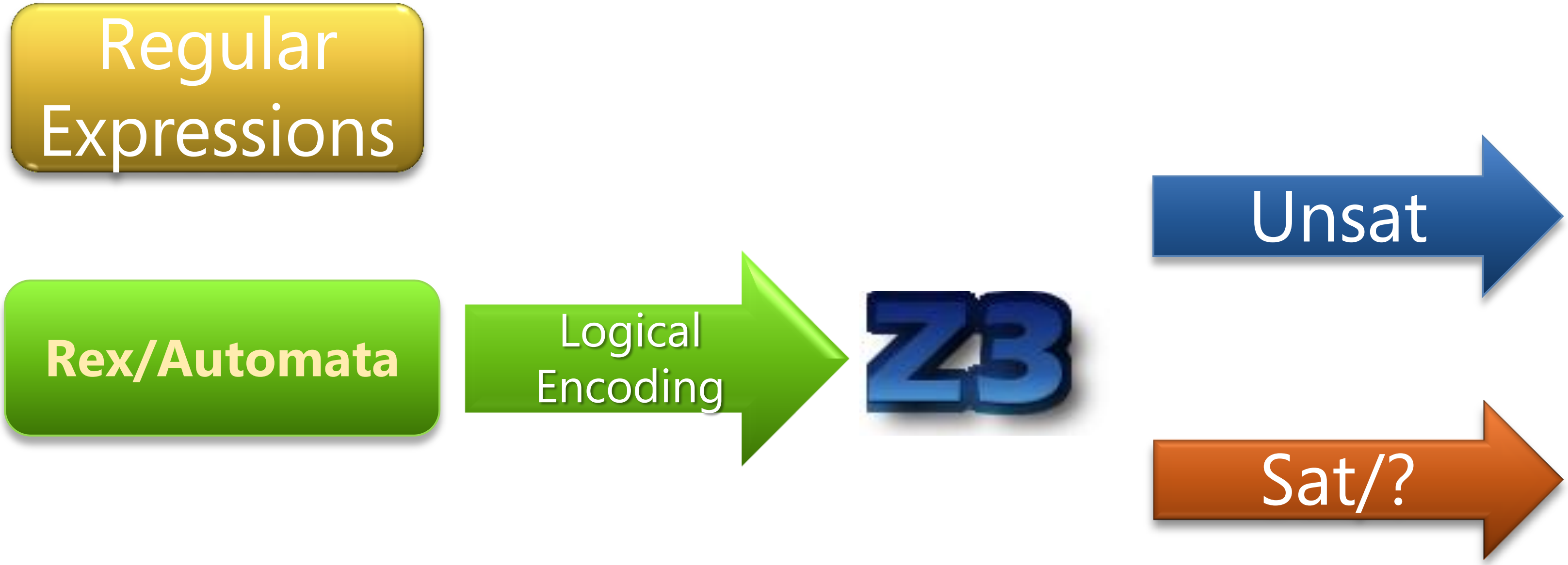
```
[b - [b]]
```

Can you discover the secret regex? Ask ***Rex****!*

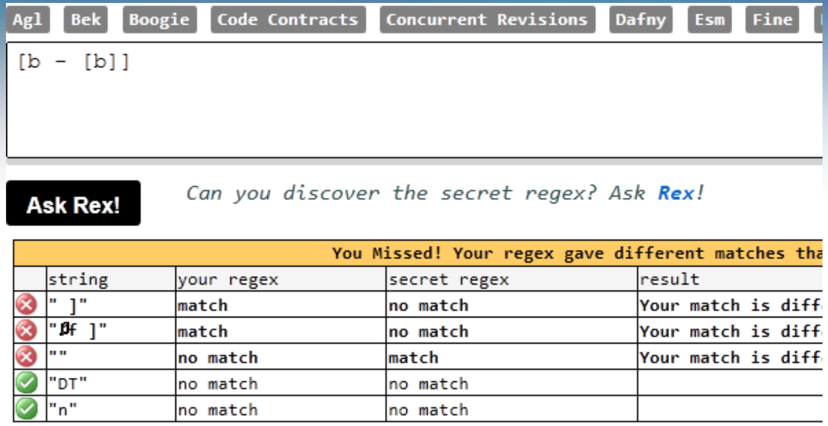
You Missed! Your regex gave different matches tha

	string	your regex	secret regex	result
☒	"]"	match	no match	Your match is diff
☒	"ßf]"	match	no match	Your match is diff
☒	""	no match	match	Your match is diff
✓	"DT"	no match	no match	
✓	"n"	no match	no match	





Usable Automatic Answers

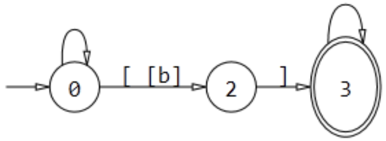


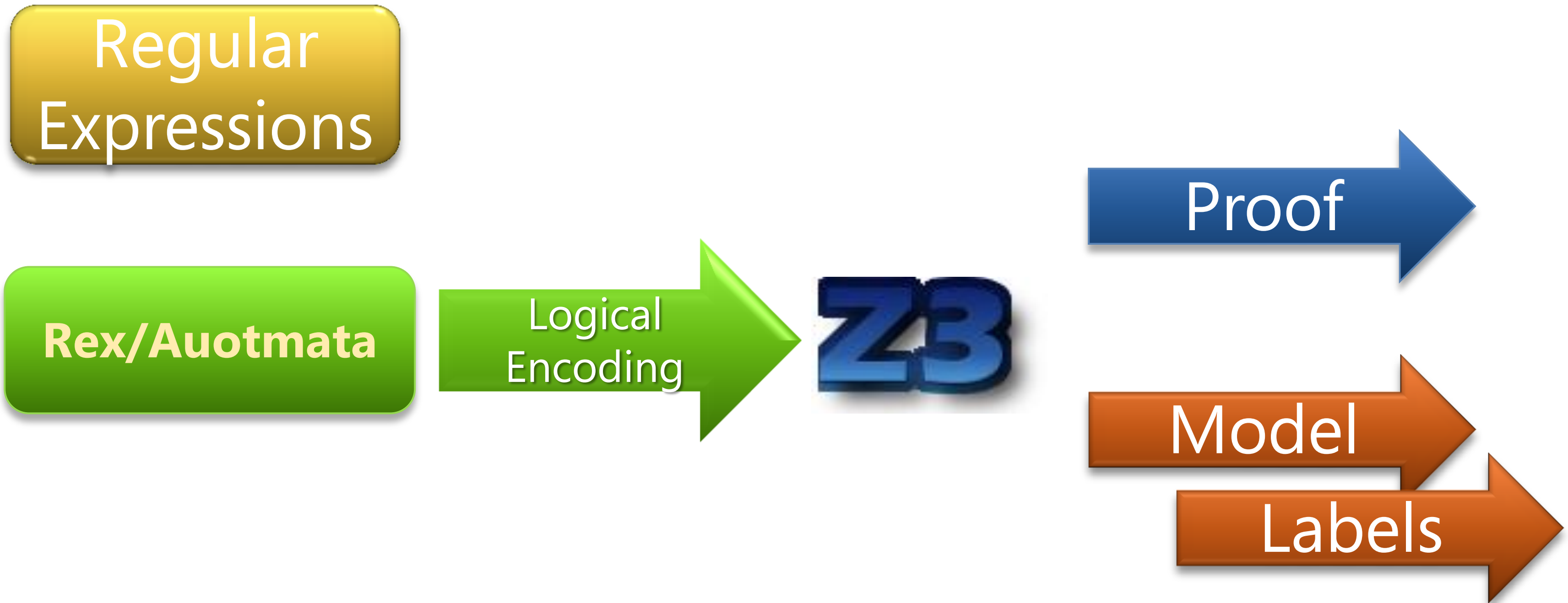
```
[b - [b]]
```

Can you discover the secret regex? Ask ***Rex****!*

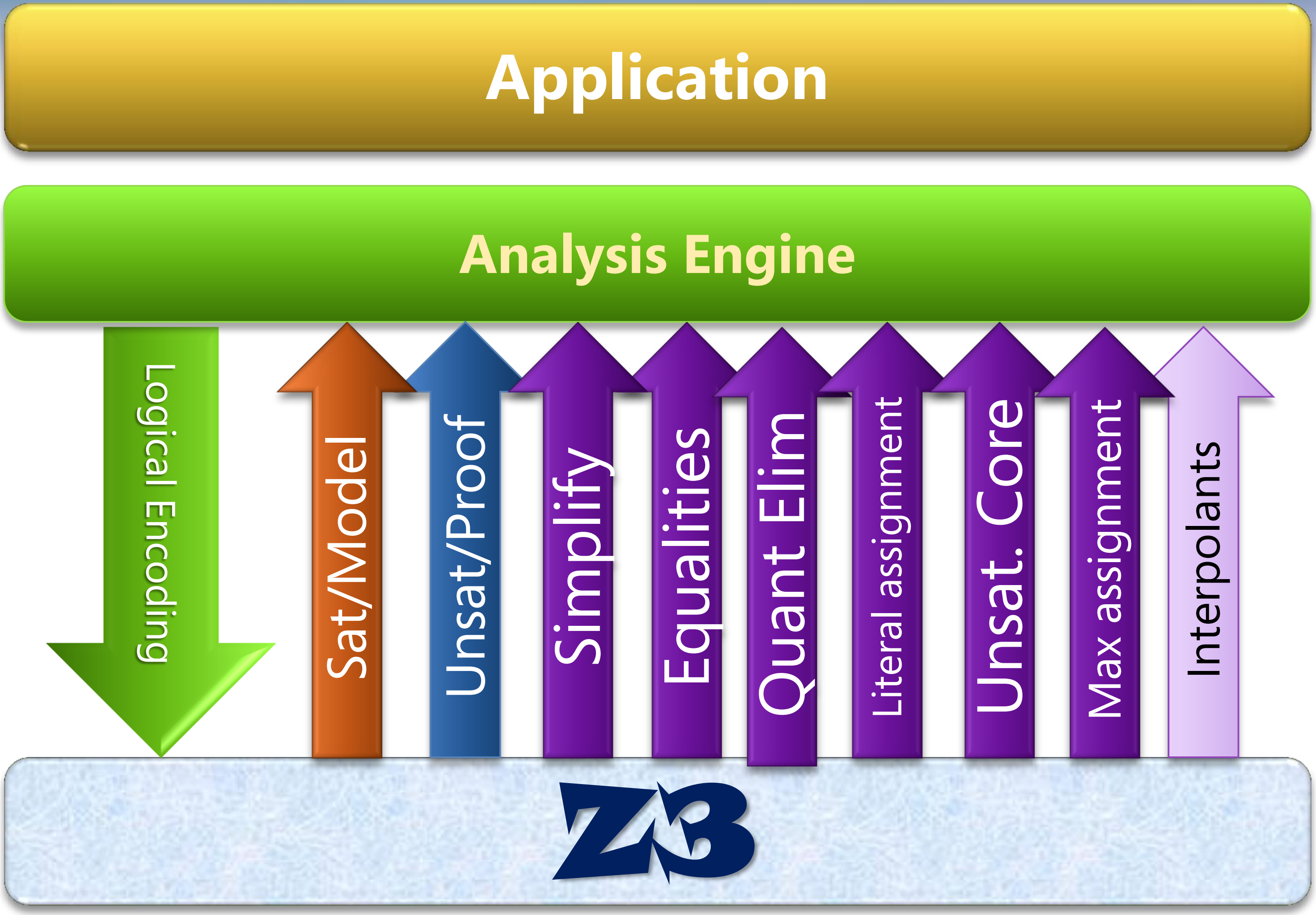
You Missed! Your regex gave different matches tha

	string	your regex	secret regex	result
✗	"]"	match	no match	Your match is diff
✗	"Øf]"	match	no match	Your match is diff
✗	""	no match	match	Your match is diff
✓	"DT"	no match	no match	
✓	"n"	no match	no match	





Usable Automatic Answers



Usable Automatic Answers

Sat/unsat answers alone have limited use

Model/Proof answers help for

- Models: Debugging during verification
- Proofs: can use solver as untrusted Oracle

Much more is possible and needed

- Many existing applications wrap several calls into solver, re-using partial information.
- Many potential applications use objective functions.

Efficiency and Expressivity

Z3 uses DPLL(T) as basic architecture.

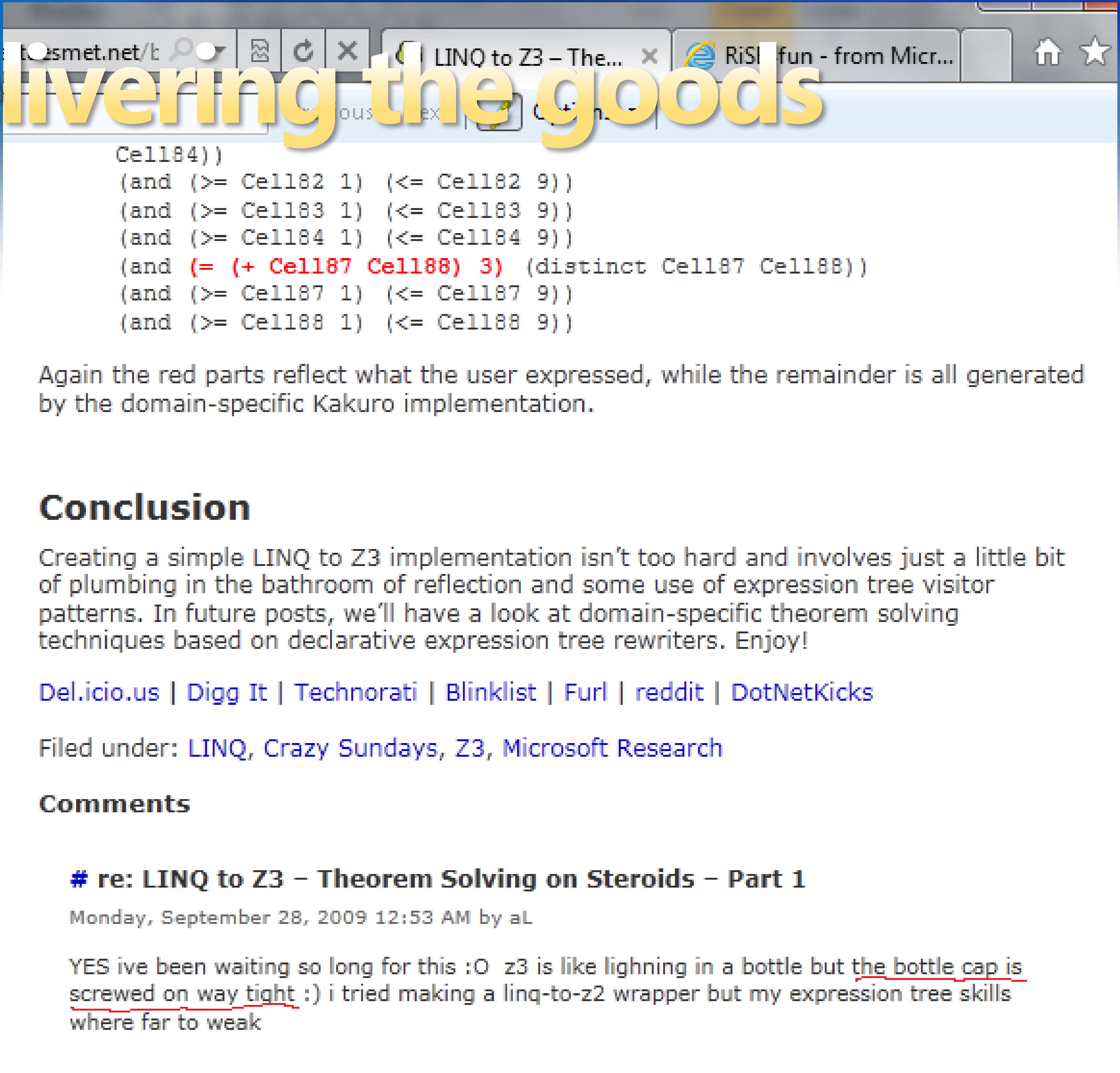
- Based on efficient DPLL for SAT solvers
- Extensible by theory solvers

DPLL(T) alone is not enough:

- DPLL(Γ) – add super-position
- DPLL(T) can be exponentially worse than unrestricted resolution.
  - DPLL(⊔) – solving diamonds
  - CDTR: Conflict Directed Theory Resolution
    Claim

    DPLL(T) + CDTR + Restart $\equiv_p$ Unrestricted T-Resolution

Delivering the goods



```
Cell184))
(and (>= Cell182 1) (<= Cell182 9))
(and (>= Cell183 1) (<= Cell183 9))
(and (>= Cell184 1) (<= Cell184 9))
(and (= (+ Cell187 Cell188) 3) (distinct Cell187 Cell188))
(and (>= Cell187 1) (<= Cell187 9))
(and (>= Cell188 1) (<= Cell188 9))
```

Again the red parts reflect what the user expressed, while the remainder is all generated by the domain-specific Kakuro implementation.

Conclusion

Creating a simple LINQ to Z3 implementation isn't too hard and involves just a little bit of plumbing in the bathroom of reflection and some use of expression tree visitor patterns. In future posts, we'll have a look at domain-specific theorem solving techniques based on declarative expression tree rewriters. Enjoy!

Del.icio.us | Digg It | Technorati | Blinklist | Furl | reddit | DotNetKicks

Filed under: LINQ, Crazy Sundays, Z3, Microsoft Research

Comments

# re: LINQ to Z3 – Theorem Solving on Steroids – Part 1

Monday, September 28, 2009 12:53 AM by aL

YES ive been waiting so long for this :O z3 is like lighning in a bottle but the bottle cap is screwed on way tight :) i tried making a linq-to-z2 wrapper but my expression tree skills where far to weak

A LINQ/F# Quotations primer

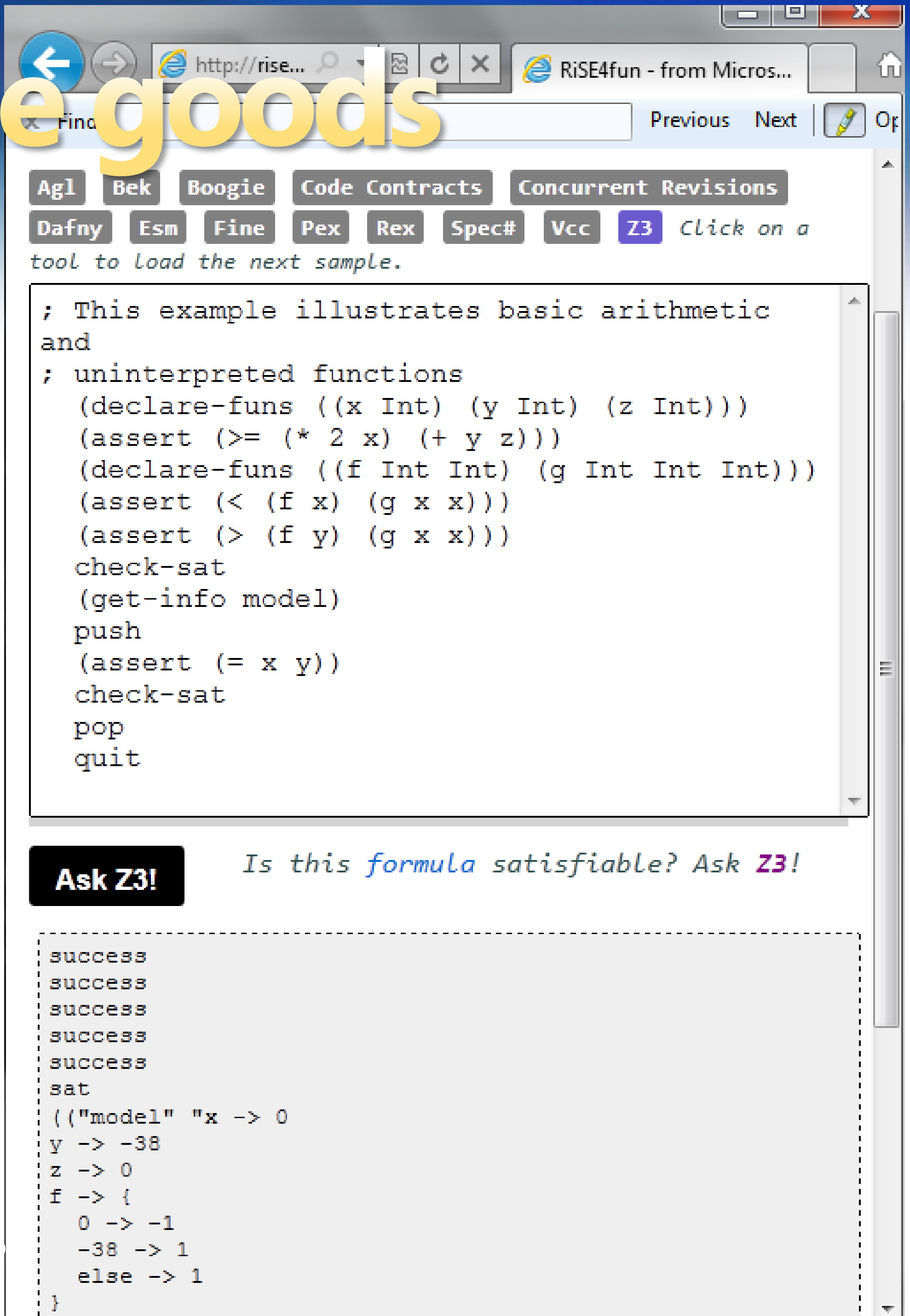
Create Quoted Expression

```
open Microsoft.Z3
open Microsoft.Z3.Quotations

 do Solver.prove <@ Logic.declare
       (fun t11 t12 t21 t22 t31 t32 ->
          not
             ((t11 >= 0I) && (t12 >= t11 + 2I) && (t12 + 1I <= 8I) &&
              (t21 >= 0I) && (t22 >= t21 + 3I) && (t32 + 1I <= 8I) &&
              (t31 >= 0I) && (t32 >= t31 + 2I) && (t32 + 3I <= 8I) &&
              (t11 >= t21 + 3I || t21 >= t11 + 2I) &&
              (t11 >= t31 + 2I || t31 >= t11 + 2I) &&
              (t21 >= t31 + 2I || t31 >= t21 + 3I) &&
              (t12 >= t22 + 1I || t22 >= t12 + 1I) &&
              (t12 >= t32 + 3I || t32 >= t12 + 1I) &&
              (t22 >= t32 + 3I || t32 >= t22 + 1I)
             )
        )
     @>
```

Delivering the goods

- 👍 No installation
- 👍 Support for SMT-LIB2 notation
- 👎 Only usable for bare bones logic encoding



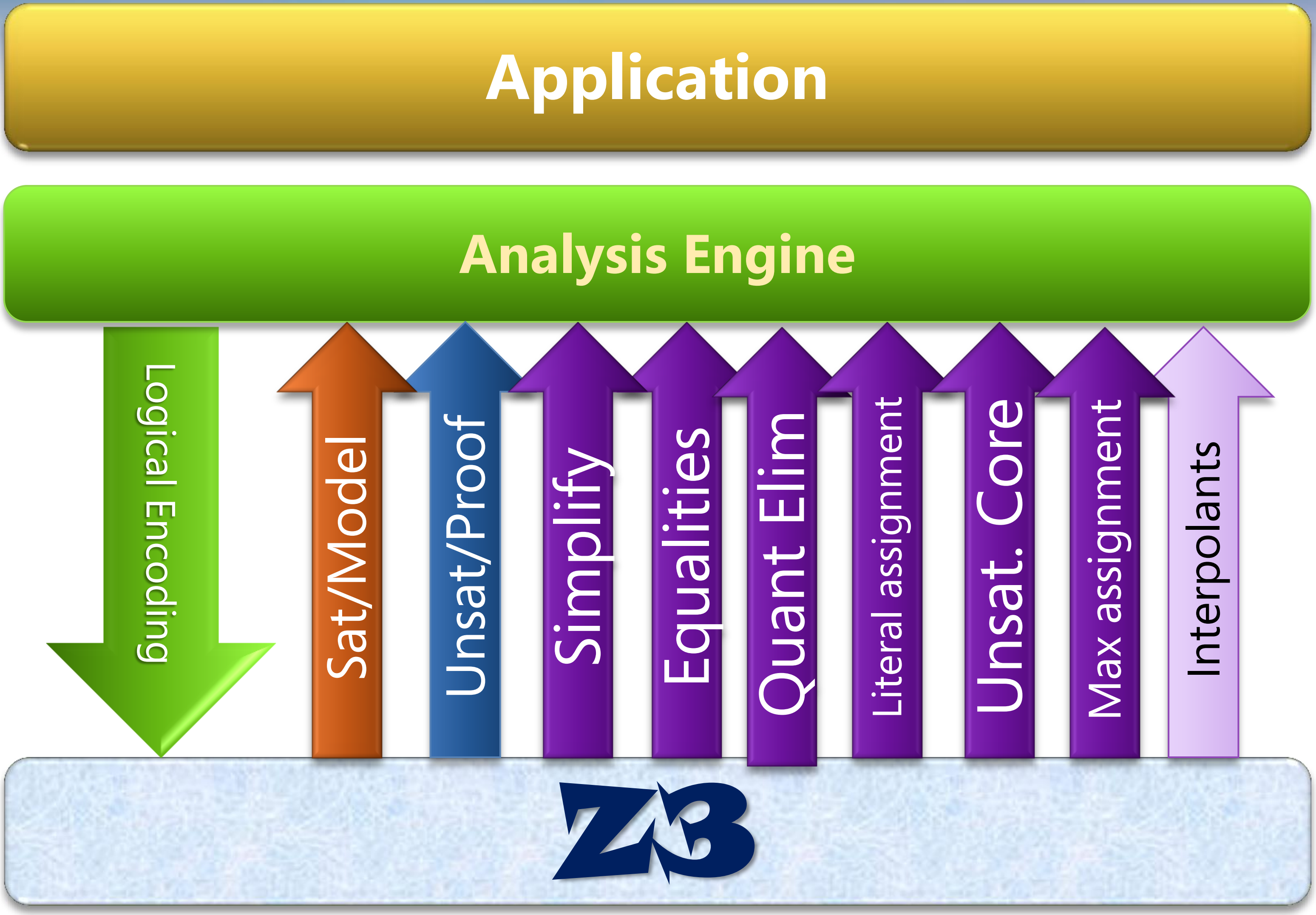
Click on a tool to load the next sample.

```
; This example illustrates basic arithmetic
and
; uninterpreted functions
   (declare-funs ((x Int) (y Int) (z Int)))
   (assert (>= (* 2 x) (+ y z)))
   (declare-funs ((f Int Int) (g Int Int Int)))
   (assert (< (f x) (g x x)))
   (assert (> (f y) (g x x)))
  check-sat
   (get-info model)
  push
   (assert (= x y))
  check-sat
  pop
  quit
```

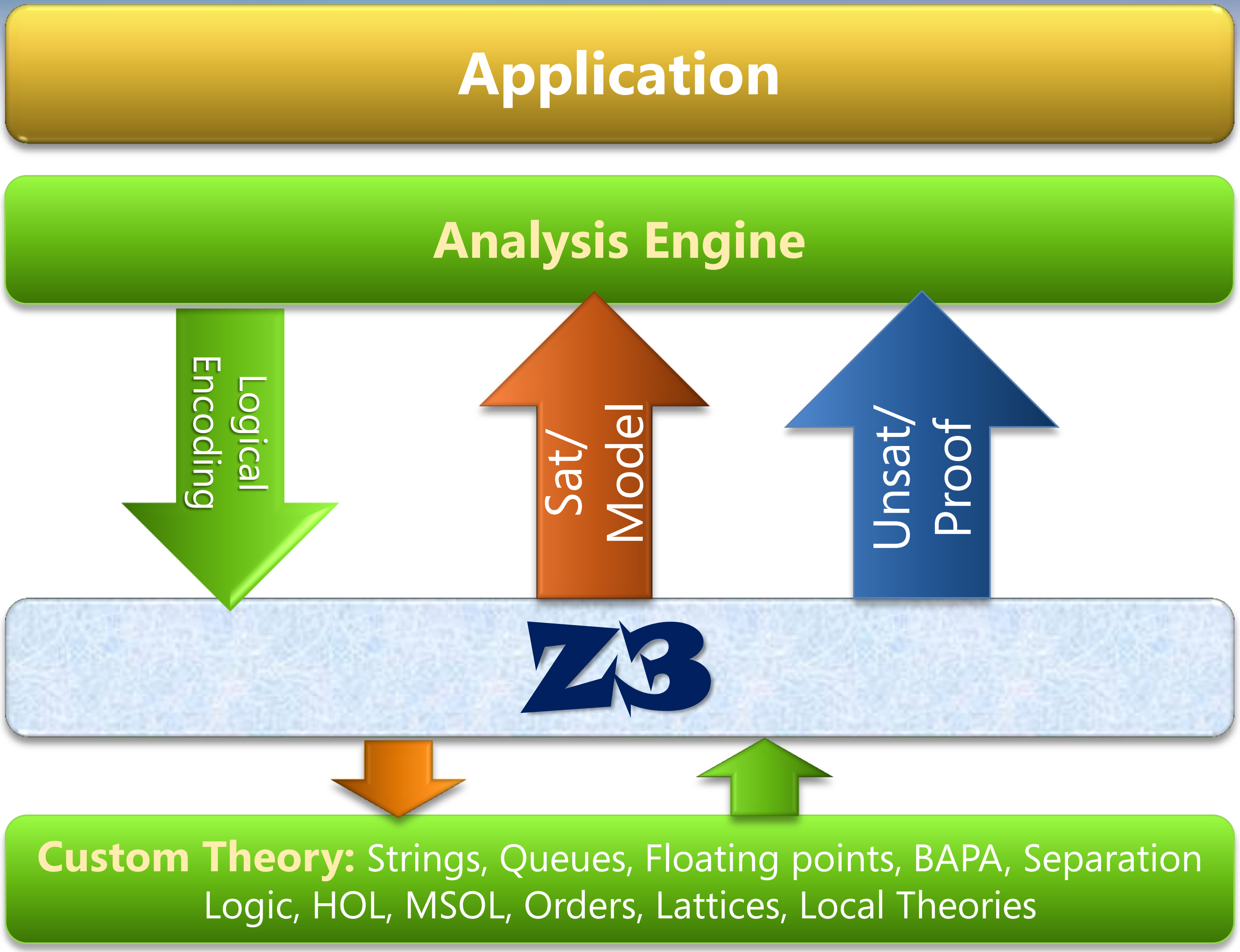
Is this formula satisfiable? Ask Z3!

```
success
success
success
success
success
sat
(("model" "x -> 0
y -> -38
z -> 0
f -> {
  0 -> -1
  -38 -> 1
  else -> 1
}
```

User Theories



User Theories



Application

Analysis Engine

Logical Encoding

Sat/ Model

Unsat/ Proof

Z3

Custom Theory: Strings, Queues, Floating points, BAPA, Separation Logic, HOL, MSOL, Orders, Lattices, Local Theories

Conclusions

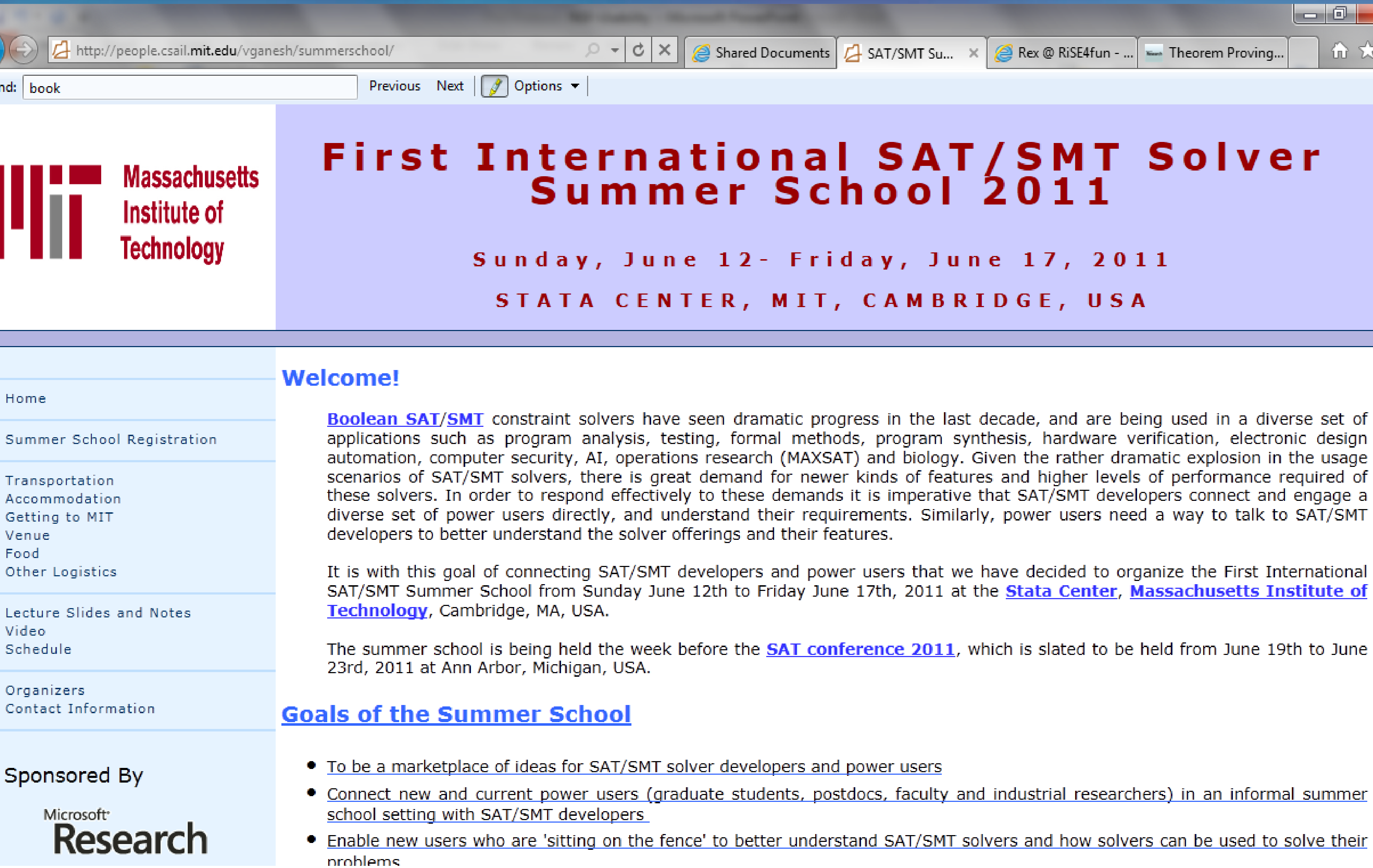
Usability addressed by:

- Lower barrier of entry for first use: http://rise4fun.com/z3
- Enable basic input formats: SMT-LIB2, C, .NET, F# Quotations, LINQ….
- Improved efficiency/scale for theory & quantifier reasoning
- Extensible by user solvers

Usability challenges:

- You too should use Z3
- Writing a user theory solver is not for the faint of heart
  - "What assumptions of the solver should I and can I make"?
  - "I would like to predict the search behavior"

Summer school on useful tools for verification



First International SAT/SMT Solver Summer School 2011

Sunday, June 12- Friday, June 17, 2011

STATA CENTER, MIT, CAMBRIDGE, USA

Welcome!

Boolean SAT/SMT constraint solvers have seen dramatic progress in the last decade, and are being used in a diverse set of applications such as program analysis, testing, formal methods, program synthesis, hardware verification, electronic design automation, computer security, AI, operations research (MAXSAT) and biology. Given the rather dramatic explosion in the usage scenarios of SAT/SMT solvers, there is great demand for newer kinds of features and higher levels of performance required of these solvers. In order to respond effectively to these demands it is imperative that SAT/SMT developers connect and engage a diverse set of power users directly, and understand their requirements. Similarly, power users need a way to talk to SAT/SMT developers to better understand the solver offerings and their features.

It is with this goal of connecting SAT/SMT developers and power users that we have decided to organize the First International SAT/SMT Summer School from Sunday June 12th to Friday June 17th, 2011 at the Stata Center, Massachusetts Institute of Technology, Cambridge, MA, USA.

The summer school is being held the week before the SAT conference 2011, which is slated to be held from June 19th to June 23rd, 2011 at Ann Arbor, Michigan, USA.

Goals of the Summer School

- To be a marketplace of ideas for SAT/SMT solver developers and power users
- Connect new and current power users (graduate students, postdocs, faculty and industrial researchers) in an informal summer school setting with SAT/SMT developers
- Enable new users who are 'sitting on the fence' to better understand SAT/SMT solvers and how solvers can be used to solve their problems

Usable Verification requires Automated Deduction!

CADE 23

Wrocław, Poland.
31 July - 5 August 2011

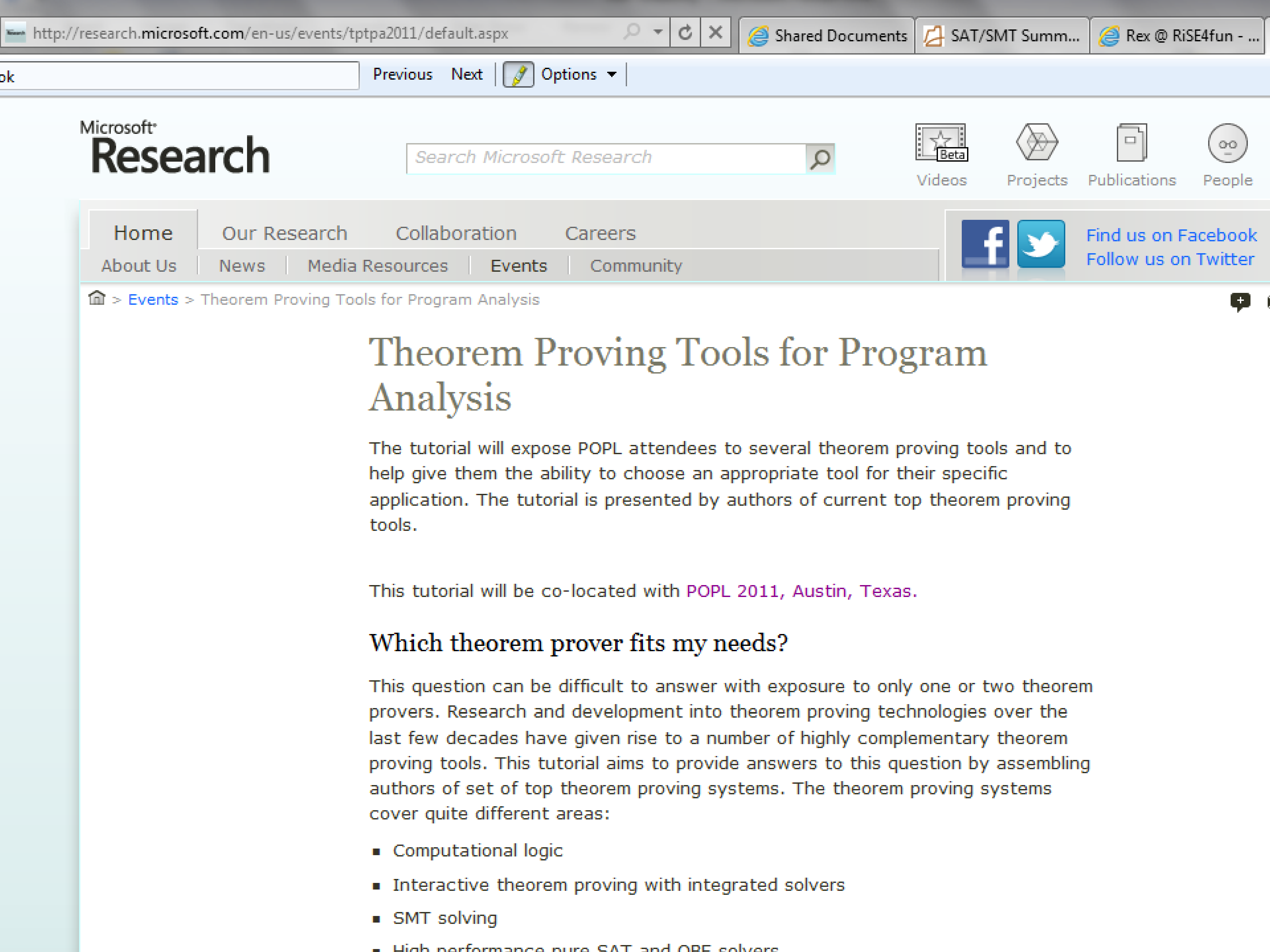
23rd international conference
on automated deduction

- **Home**
- **Call for Workshops and Tutorials**
- **Call for Papers**
- **Invited Speakers**
- **PC + Organizers**
- **About Poland**
- **Travel Information**
- **Accomodation**

Welcome to the CADE website!

Posted on 28 October 2010 by admin

CADE – the 23rd International Conference on Automated Deduction



Theorem Proving Tools for Program Analysis

The tutorial will expose POPL attendees to several theorem proving tools and to help give them the ability to choose an appropriate tool for their specific application. The tutorial is presented by authors of current top theorem proving tools.

This tutorial will be co-located with POPL 2011, Austin, Texas.

Which theorem prover fits my needs?

This question can be difficult to answer with exposure to only one or two theorem provers. Research and development into theorem proving technologies over the last few decades have given rise to a number of highly complementary theorem proving tools. This tutorial aims to provide answers to this question by assembling authors of set of top theorem proving systems. The theorem proving systems cover quite different areas:

- Computational logic
- Interactive theorem proving with integrated solvers
- SMT solving
- High performance pure SAT and QBF solvers