# *Orchestrating Decision Engines*

## University of Verona, 2011

Leonardo de Moura and Grant Passmore

# Satisfiability Modulo Theories (SMT)

A Satisfiability Checker

with built-in support for useful theories

# Satisfiability Modulo Theories (SMT)

$$b + 2 = c \;\; and \;\; f(read(write(a,b,3), c-2) \neq f(c-b+1)$$

# Satisfiability Modulo Theories (SMT)

$b + 2 = c$ *and* $f(read(write(a,b,3), c-2) \neq f(c-b+1)$

Arithmetic

# Satisfiability Modulo Theories (SMT)

$$b + 2 = c \text{ and } f(read(write(a,b,3), c-2)) \neq f(c-b+1)$$

Array Theory

Microsoft Research

# Satisfiability Modulo Theories (SMT)

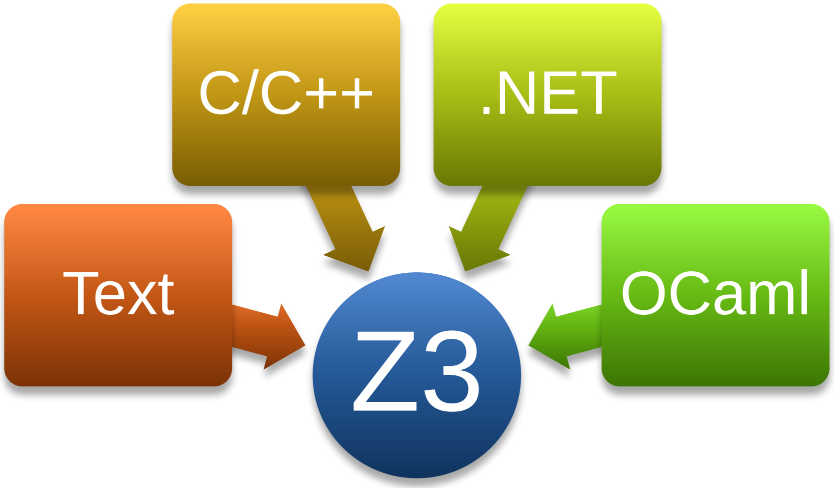
$$b + 2 = c \text{ and } f(read(write(a,b,3), c\text{-}2) \neq f(c\text{-}b+1)$$

Uninterpreted Functions

Microsoft® Research

# SMT@Microsoft: Solver

- Z3 is a solver developed at Microsoft Research.
- Development/Research driven by internal customers.
- Free for non-commercial use.
- Interfaces:




- http://research.microsoft.com/projects/z3

# Some Applications @ Microsoft

The Spec# Programming System

HAVOC

ForμLa

Hyper-V
Microsoft | Virtualization

Terminator T-2

VCC

SLAM

Yogi

NModel

Vigilante

SpecExplorer

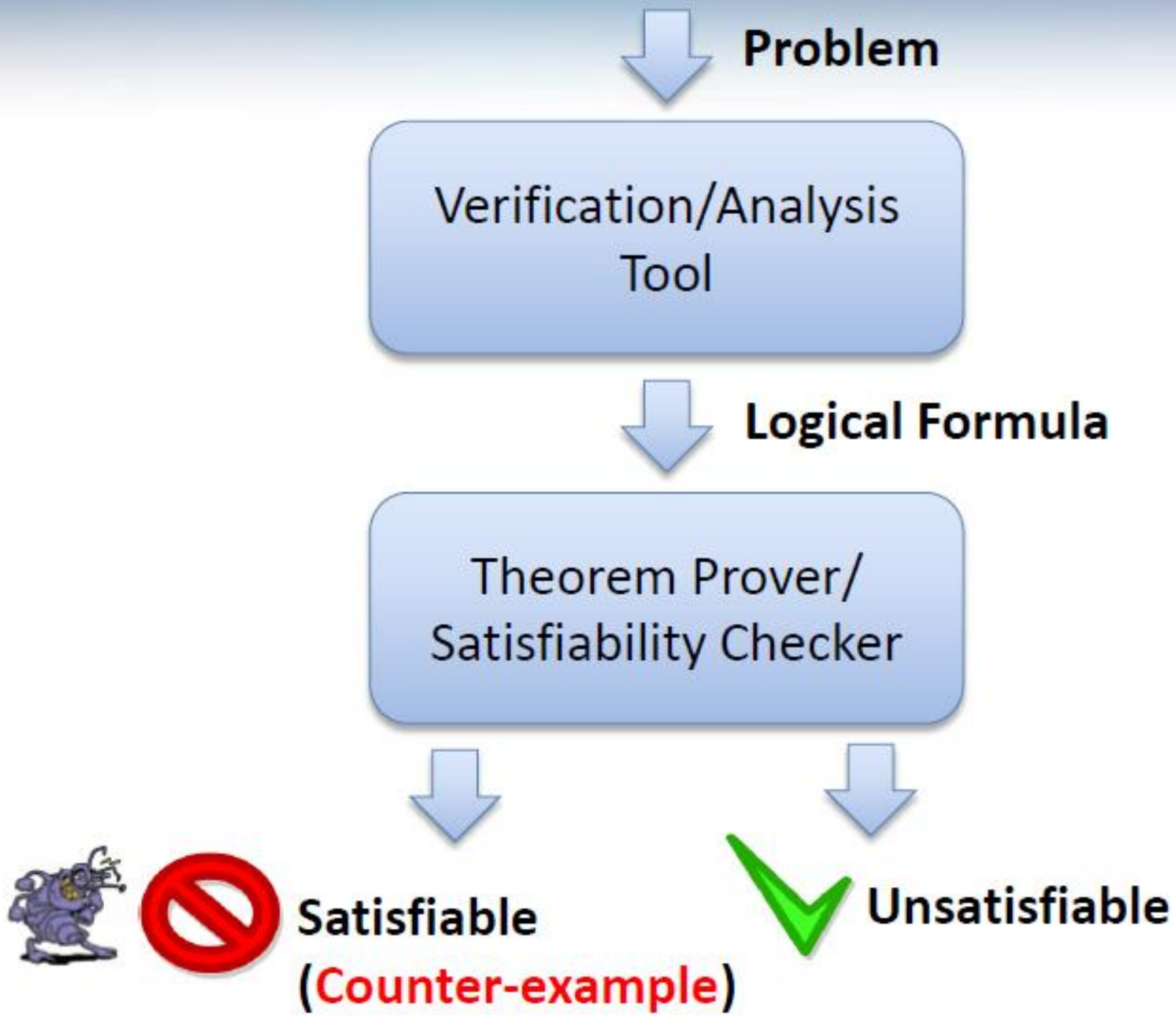
Pex

F7

SAGE

Microsoft Research

# Verification/Analysis Tool: “Template”




Microsoft Research

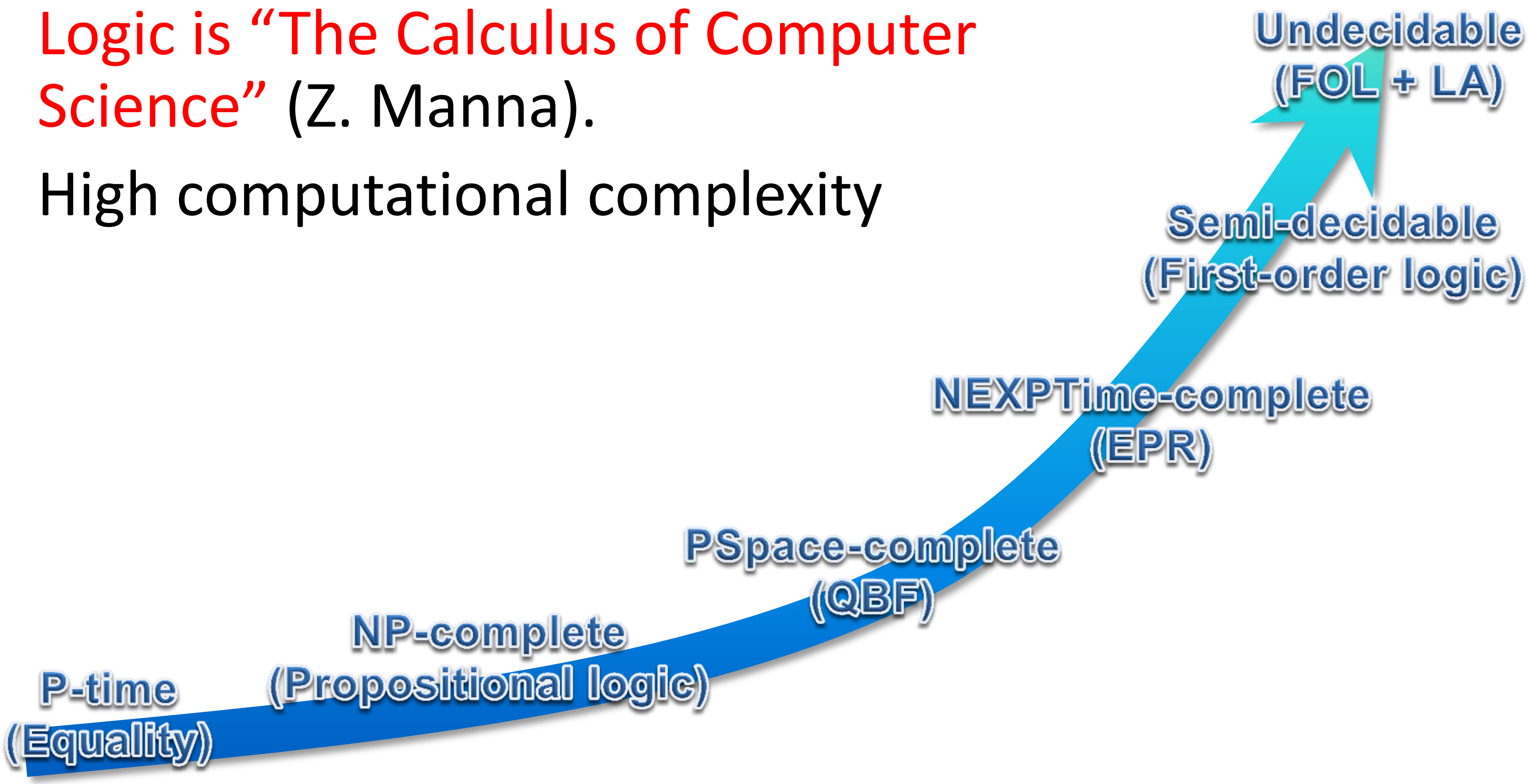
# Symbolic Reasoning

Verification/Analysis tools need some form of **Symbolic Reasoning**

Microsoft Research

# Symbolic Reasoning

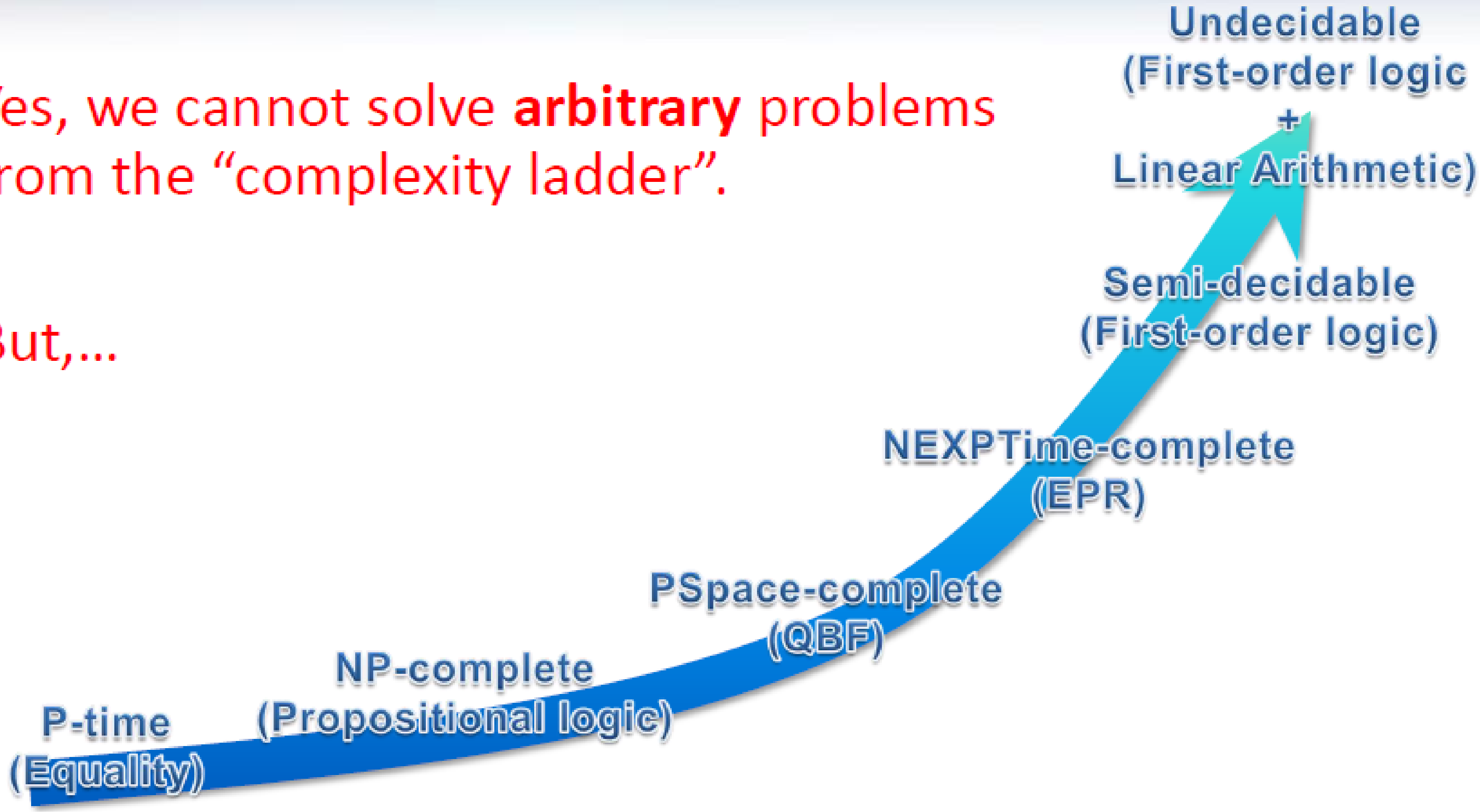
- Logic is “The Calculus of Computer Science” (Z. Manna).
- High computational complexity



# Symbolic Reasoning

Yes, we cannot solve **arbitrary** problems from the "complexity ladder".

But,...



# Symbolic Reasoning

We can try to solve the problems we find in

**real applications**

# Applications

- Test case generation
- Verifying Compilers
- Predicate Abstraction
- Invariant Generation
- Type Checking
- Model Based Testing

Microsoft Research

# Test-case generation

- Test (correctness + usability) is 95% of the deal:
  - Dev/Test is 1-1 in products.
  - Developers are responsible for unit tests.
- Tools:
  - Annotations and static analysis (SAL + ESP)
  - File Fuzzing
  - Unit test case generation

Microsoft Research

# Security is critical

- Security bugs can be very expensive:
  - Cost of each MS Security Bulletin: $600k to $Millions.
  - Cost due to worms: $Billions.
  - The real victim is the customer.
- Most security exploits are initiated via files or packets.
  - Ex: Internet Explorer parses dozens of file formats.
- Security testing: hunting for million dollar bugs
  - Write A/V
  - Read A/V
  - Null pointer dereference
  - Division by zero

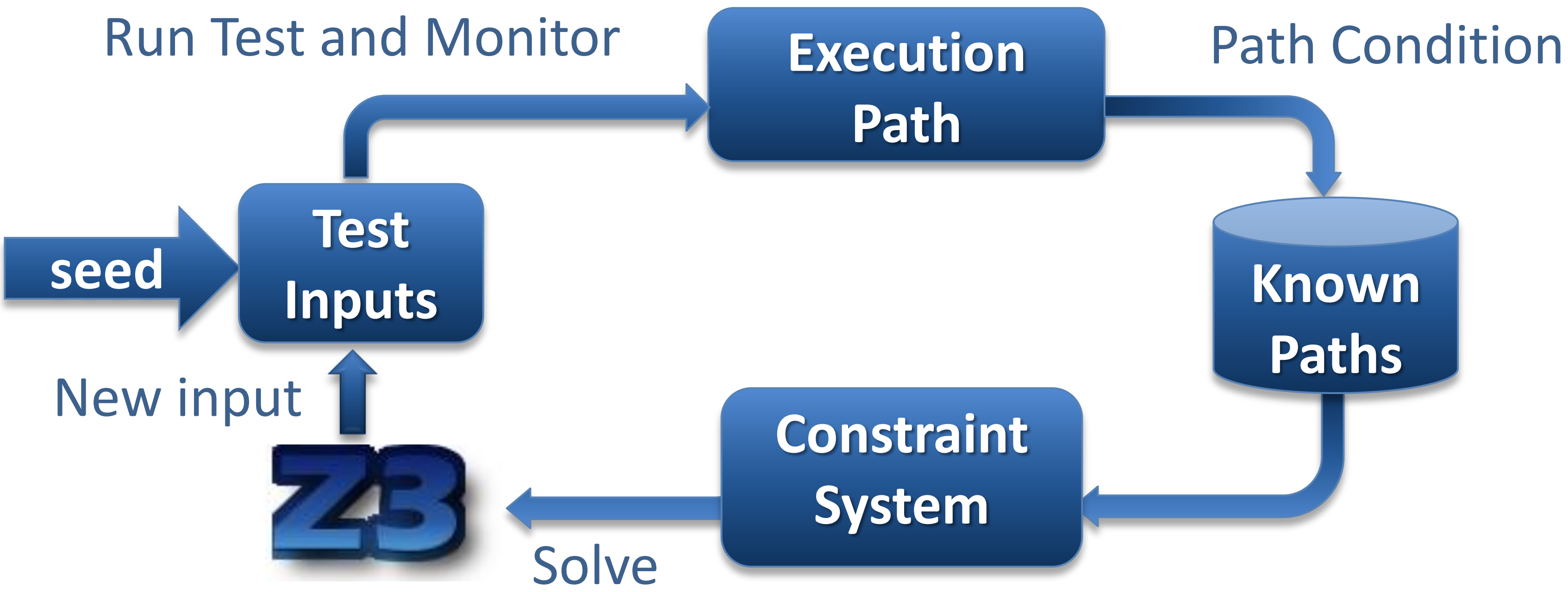
Microsoft Research

# Hunting for Security Bugs.

- Two main techniques used by *"black hats"*:
  - Code inspection (of binaries).
  - *Black box fuzz testing.*

- **Black box** fuzz testing:
  - A form of black box random testing.
  - Randomly *fuzz* (=modify) a well formed input.
  - Grammar-based fuzzing: rules to encode how to fuzz.
- **Heavily** used in security testing
  - At MS: several internal tools.
  - Conceptually simple yet effective in practice

# Directed Automated Random Testing ( DART)




Microsoft Research

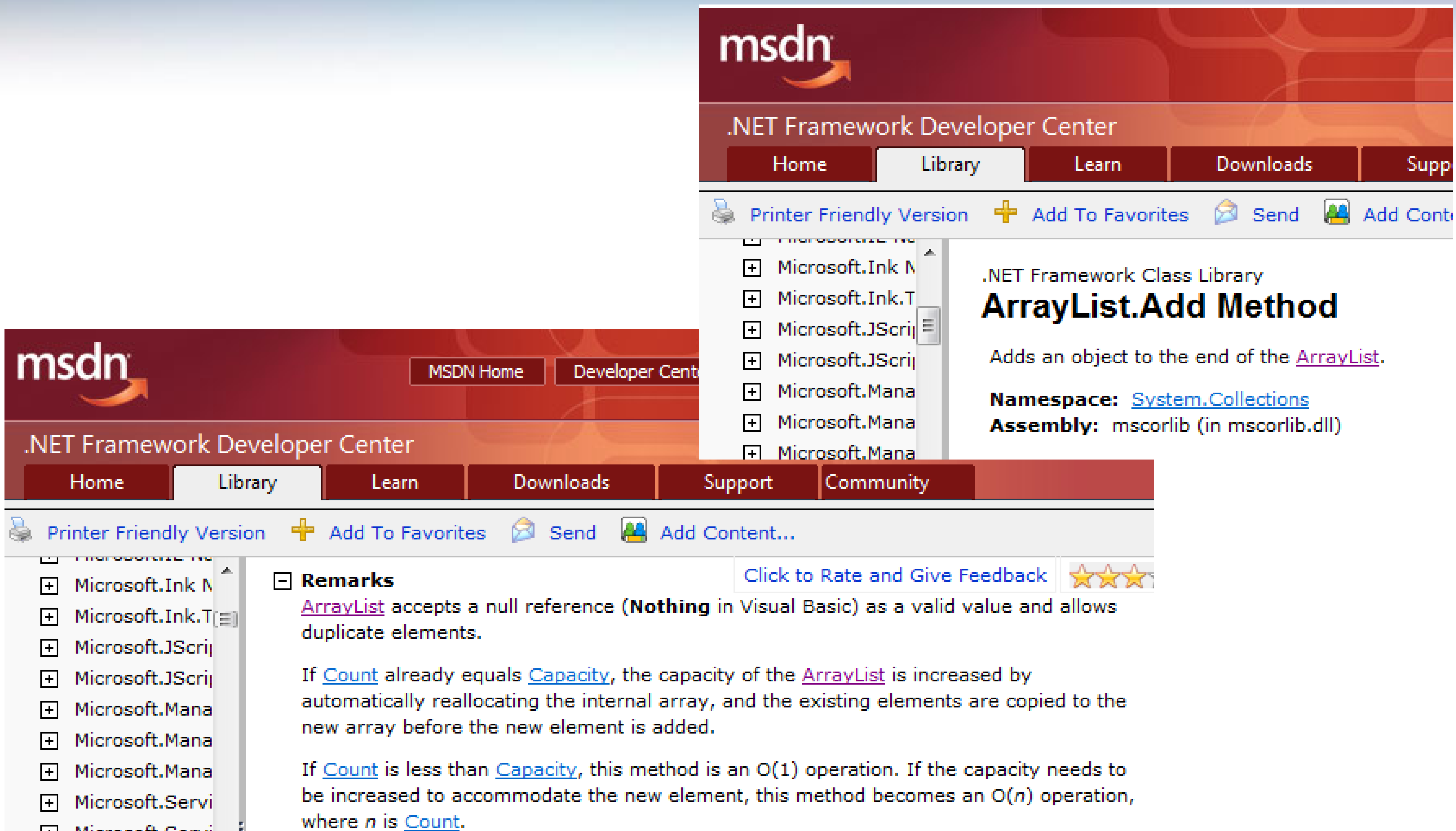
# DARTish projects at Microsoft

- **PEX** — Implements DART for .NET.
- **SAGE** — Implements DART for x86 binaries.
- **YOGI** — Implements DART to check the feasibility of program paths generated statically using a SLAM-like tool.
- **Vigilante** — Partially implements DART to dynamically generate worm filters.

# What is Pex?

- Test input generator
  - Pex starts from parameterized unit tests
  - Generated tests are emitted as traditional unit tests

# ArrayList: The Spec

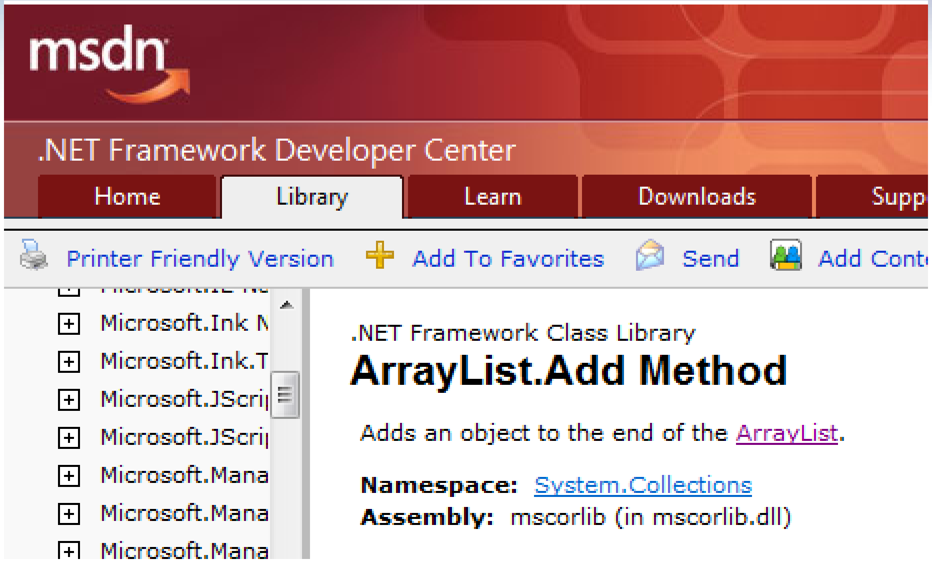



Microsoft Research

# ArrayList: AddItem Test

```
class ArrayListTest {
  [PexMethod]
  void AddItem(int c, object item) {
      var list = new ArrayList(c);
      list.Add(item);
      Assert(list[0] == item); }
}
```

```
class ArrayList {
  object[] items;
  int count;

  ArrayList(int capacity) {
    if (capacity < 0) throw ...;
    items = new object[capacity];
  }

  void Add(object item) {
    if (count == items.Length)
      ResizeArray();

    items[this.count++] = item; }
...
```




Microsoft Research

# ArrayList: Starting Pex…

```
class ArrayListTest {
  [PexMethod]
  void AddItem(int c, object item) {
      var list = new ArrayList(c);
      list.Add(item);
      Assert(list[0] == item); }
}
```

| Inputs |
|---|
| |

```
class ArrayList {
  object[] items;
  int count;

  ArrayList(int capacity) {
    if (capacity < 0) throw ...;
    items = new object[capacity];
  }

  void Add(object item) {
    if (count == items.Length)
      ResizeArray();

    items[this.count++] = item; }
...
```

Microsoft Research

# ArrayList: Run 1, (0,null)

```
class ArrayListTest {
  [PexMethod]
  void AddItem(int c, object item) {
      var list = new ArrayList(c);
      list.Add(item);
      Assert(list[0] == item); }
}
```

```
class ArrayList {
  object[] items;
  int count;

  ArrayList(int capacity) {
    if (capacity < 0) throw ...;
    items = new object[capacity];
  }

  void Add(object item) {
    if (count == items.Length)
      ResizeArray();

    items[this.count++] = item; }
...
```

| Inputs |
| --- |
| (0,null) |

# ArrayList: Run 1, (0,null)

```
class ArrayListTest {
  [PexMethod]
  void AddItem(int c, object item) {
      var list = new ArrayList(c);
      list.Add(item);
      Assert(list[0] == item); }
}
```

| | Inputs | Observed Constraints |
|---|---|---|
| | (0,null) | !(c<0) |

```
class ArrayList {
  object[] items;
  int count;

  ArrayList(int capacity) {
    if (capacity < 0) throw ...;
    items = new object[capacity];
  }

  void Add(object item) {
    if (count == items.Length)
      ResizeArray();

    items[this.count++] = item; }
...
```

c < 0 → false

Microsoft Research

# ArrayList: Run 1, (0,null)

```
class ArrayListTest {
  [PexMethod]
  void AddItem(int c, object item) {
      var list = new ArrayList(c);
      list.Add(item);
      Assert(list[0] == item); }
}
```

| | Inputs | Observed Constraints |
|---|---|---|
| | (0,null) | !(c<0) && 0==c |

```
class ArrayList {
  object[] items;
  int count;

  ArrayList(int capacity) {
    if (capacity < 0) throw ...;
    items = new object[capacity];
  }

  void Add(object item) {
    if (count == items.Length)
      ResizeArray();

    items[this.count++] = item; }
...
```

0 == c → true

# ArrayList: Run 1, (0,null)

```
class ArrayListTest {
  [PexMethod]
  void AddItem(int c, object item) {
      var list = new ArrayList(c);
      list.Add(item);
      Assert(list[0] == item); }
}
```

| | Inputs | Observed Constraints |
|---|---|---|
| | (0,null) | !(c<0) && 0==c |

```
item == item  →  true
```

```
class ArrayList {
  object[] items;
  int count;

  ArrayList(int capacity) {
    if (capacity < 0) throw ...;
    items = new object[capacity];
  }

  void Add(object item) {
    if (count == items.Length)
      ResizeArray();

    items[this.count++] = item; }
...
```

Microsoft Research

# ArrayList: Picking the next branch to cover

```
class ArrayListTest {
  [PexMethod]
  void AddItem(int c, object item) {
      var list = new ArrayList(c);
      list.Add(item);
      Assert(list[0] == item); }
}
```

```
class ArrayList {
  object[] items;
  int count;

  ArrayList(int capacity) {
    if (capacity < 0) throw ...;
    items = new object[capacity];
  }

  void Add(object item) {
    if (count == items.Length)
      ResizeArray();

    items[this.count++] = item; }
...
```

| Constraints to solve | Inputs | Observed Constraints |
|---|---|---|
| | (0,null) | !(c<0) && 0==c |
| !(c<0) && **0!=c** | | |

# ArrayList: Solve constraints using SMT solver

```
class ArrayListTest {
  [PexMethod]
  void AddItem(int c, object item) {
      var list = new ArrayList(c);
      list.Add(item);
      Assert(list[0] == item); }
}
```

```
class ArrayList {
  object[] items;
  int count;

  ArrayList(int capacity) {
    if (capacity < 0) throw ...;
    items = new object[capacity];
  }

  void Add(object item) {
    if (count == items.Length)
      ResizeArray();

    items[this.count++] = item; }
...
```

| Constraints to solve | Inputs | Observed Constraints |
|---|---|---|
| | (0,null) | !(c<0) && 0==c |
| !(c<0) && 0!=c | **(1,null)** | |


Microsoft Research

# ArrayList: Run 2, (1, null)

```
class ArrayListTest {
  [PexMethod]
  void AddItem(int c, object item) {
      var list = new ArrayList(c);
      list.Add(item);
      Assert(list[0] == item); }
}
```

| Constraints to solve | Inputs | Observed Constraints |
|---|---|---|
| | (0,null) | !(c<0) && 0==c |
| !(c<0) && 0!=c | (1,null) | !(c<0) && 0!=c |

```
class ArrayList {
  object[] items;
  int count;

  ArrayList(int capacity) {
    if (capacity < 0) throw ...;
    items = new object[capacity];
  }

  void Add(object item) {
    if (count == items.Length)
      ResizeArray();

    items[this.count++] = item; }
...
```

0 == c → false

Microsoft Research

# ArrayList: Pick new branch

```
class ArrayListTest {
  [PexMethod]
  void AddItem(int c, object item) {
      var list = new ArrayList(c);
      list.Add(item);
      Assert(list[0] == item); }
}
```

```
class ArrayList {
  object[] items;
  int count;

  ArrayList(int capacity) {
    if (capacity < 0) throw ...;
    items = new object[capacity];
  }

  void Add(object item) {
    if (count == items.Length)
      ResizeArray();

    items[this.count++] = item; }
...
```

| Constraints to solve | Inputs | Observed Constraints |
|---|---|---|
| | (0,null) | !(c<0) && 0==c |
| !(c<0) && 0!=c | (1,null) | !(c<0) && 0!=c |
| c<0 | | |

Z3

Microsoft Research

# ArrayList: Run 3, (-1, null)

```
class ArrayListTest {
  [PexMethod]
  void AddItem(int c, object item) {
      var list = new ArrayList(c);
      list.Add(item);
      Assert(list[0] == item); }
}
```

```
class ArrayList {
  object[] items;
  int count;

  ArrayList(int capacity) {
    if (capacity < 0) throw ...;
    items = new object[capacity];
  }

  void Add(object item) {
    if (count == items.Length)
      ResizeArray();

    items[this.count++] = item; }
...
```

| Constraints to solve | Inputs | Observed Constraints |
|---|---|---|
| | (0,null) | !(c<0) && 0==c |
| !(c<0) && 0!=c | (1,null) | !(c<0) && 0!=c |
| c<0 | **(-1,null)** | |

# ArrayList: Run 3, (-1, null)

```
class ArrayListTest {
  [PexMethod]
  void AddItem(int c, object item) {
      var list = new ArrayList(c);
      list.Add(item);
      Assert(list[0] == item); }
}
```

| Constraints to solve | Inputs | Observed Constraints |
|---|---|---|
| | (0,null) | !(c<0) && 0==c |
| !(c<0) && 0!=c | (1,null) | !(c<0) && 0!=c |
| c<0 | **(-1,null)** | c<0 |

```
class ArrayList {
  object[] items;
  int count;

  ArrayList(int capacity) {
    if (capacity < 0) throw ...;
    items = new object[capacity];
  }

  void Add(object item) {
    if (count == items.Length)
      ResizeArray();

    items[this.count++] = item; }
...
```

c < 0 → true

Microsoft Research

# ArrayList: Run 3, (-1, null)

```
class ArrayListTest {
  [PexMethod]
  void AddItem(int c, object item) {
      var list = new ArrayList(c);
      list.Add(item);
      Assert(list[0] == item); }
}
```

```
class ArrayList {
  object[] items;
  int count;

  ArrayList(int capacity) {
    if (capacity < 0) throw ...;
    items = new object[capacity];
  }

  void Add(object item) {
    if (count == items.Length)
      ResizeArray();

    items[this.count++] = item; }
...
```

| Constraints to solve | Inputs | Observed Constraints |
|---|---|---|
| | (0,null) | !(c<0) && 0==c |
| !(c<0) && 0!=c | (1,null) | !(c<0) && 0!=c |
| c<0 | (-1,null) | c<0 |

Microsoft Research

# ArrayList: Run 3, (-1, null)

```
class ArrayListTest {
  [PexMethod]
  void AddItem(int c, object item) {
      var list = new ArrayList(c);
      list.Add(item);
      Assert(list[0] == item); }
}
```

| Constraints to solve | Inputs | Observed Constraints |
|---|---|---|
| | (0,null) | !(c<0) && 0==c |
| !(c<0) && 0!=c | (1,null) | !(c<0) && 0!=c |
| c<0 | (-1,null) | c<0 |

```
class ArrayList {
  object[] items;
  int count;

  ArrayList(int capacity) {
    if (capacity < 0) throw ...;
    items = new object[capacity];
  }

  void Add(object item) {
    if (count == items.Length)
      ResizeArray();

    items[this.count++] = item; }
...
```


Microsoft Research

# White box testing in practice

## How to test this code?

(Real code from .NET base class libraries.)

```
[SecurityPermissionAttribute(SecurityAction.LinkDemand, Flags=SecurityPermissionFlag.SerializationFormatter)]
public ResourceReader(Stream stream)
{
    if (stream==null)
        throw new ArgumentNullException("stream");
    if (!stream.CanRead)
        throw new ArgumentException(Environment.GetResourceString("Argument_StreamNotReadable"));

    _resCache = new Dictionary<String, ResourceLocator>(FastResourceComparer.Default);
    _store = new BinaryReader(stream, Encoding.UTF8);
    // We have a faster code path for reading resource files from an assembly.
    _ums = stream as UnmanagedMemoryStream;

    BCLDebug.Log("RESMGRFILEFORMAT", "ResourceReader .ctor(Stream).  UnmanagedMemoryStream: "+(_ums!=null));
    ReadResources();
}
```

# White box testing in practice

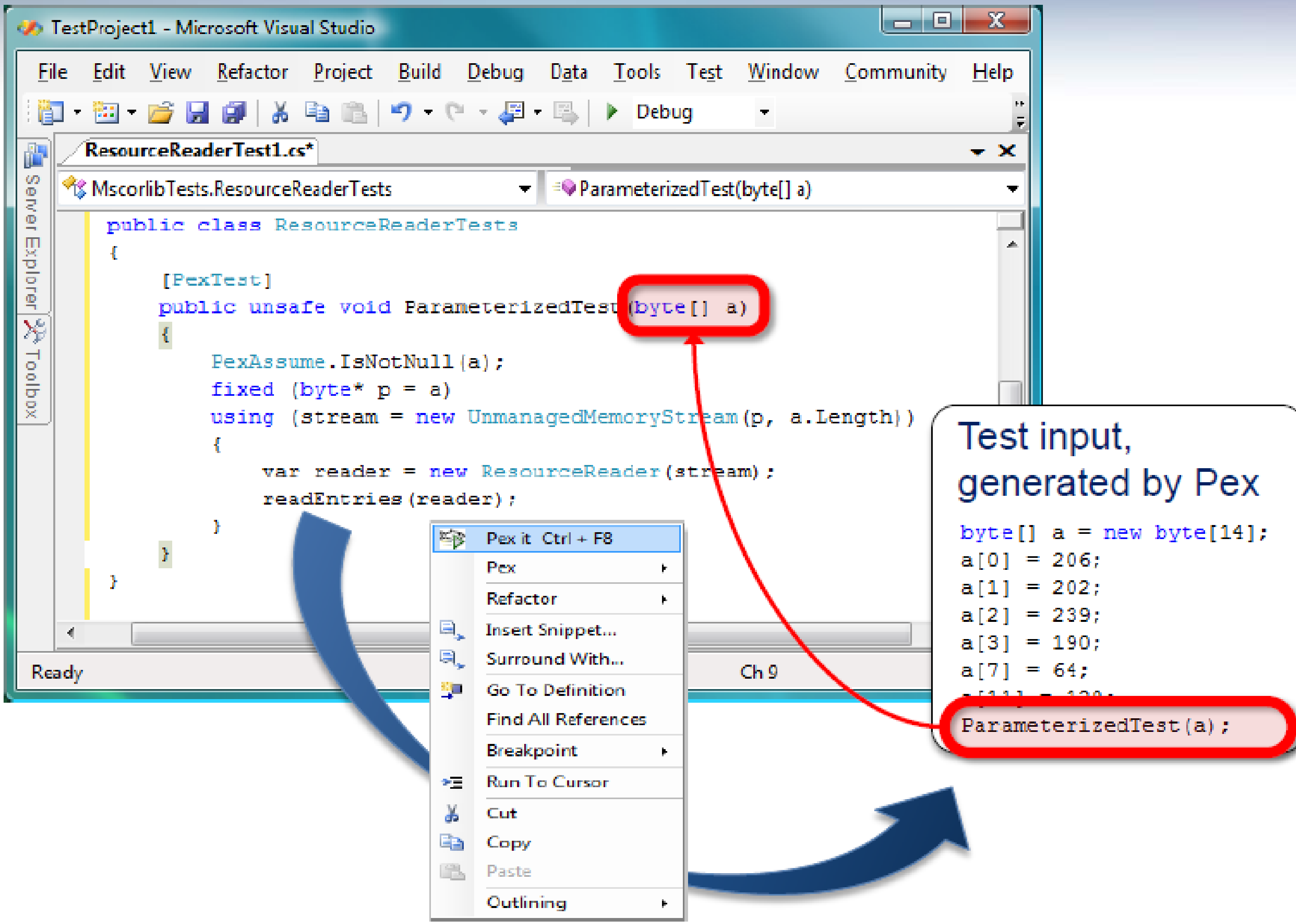
```
        // Reads in the header information for a .resources file.  Verifies some
        // of the assumptions about this resource set, and builds the class table
        // for the default resource file format.
        private void ReadResources() {
            BCLDebug.Assert(_store != null, "ResourceReader is closed!");
            BinaryFormatter bf = new BinaryFormatter(null, new StreamingContext(StreamingContextStates.File |
#if !FEATURE_PAL
            _typeLimitingBinder = new TypeLimitingDeserializationBinder();
            bf.Binder = _typeLimitingBinder;
#endif
            _objFormatter = bf;
            try {
                // Read ResourceManager header
                // Check for magic number
                int magicNum = _store.ReadInt32();
                if (magicNum != ResourceManager.MagicNumber)
                    throw new ArgumentException(Environment.GetResourceString("Resources_StreamNotValid"));
                // Assuming this is ResourceManager header V1 or greater, hopefully
                // after the version number there is a number of bytes to skip
                // to bypass the rest of the ResMgr header.
                int resMgrHeaderVersion = _store.ReadInt32();
                if (resMgrHeaderVersion > 1) {
                    int numBytesToSkip = _store.ReadInt32();
                    BCLDebug.Log("RESMGRFILEFORMAT", LogLevel.Status, "ReadResources: Unexpected ResMgr header
                    BCLDebug.Assert(numBytesToSkip >= 0, "numBytesToSkip in ResMgr header should be positive!"
                    _store.BaseStream.Seek(numBytesToSkip, SeekOrigin.Current);
                } else {
                    BCLDebug.Log("RESMGRFILEFORMAT", "ReadResources: Parsing ResMgr header v1.");
                    SkipInt32();    // We don't care about numBytesToSkip.
                    // Read in type name for a suitable ResourceReader
```

# White box testing in practice

```
        // Reads in the header information for a .resources file.  Verifies some
        // of the assumptions about this resource set, and builds the class table
        // for the default resource file format.
        private void ReadResources() {
            BCLDebug.Assert(_store != null, "ResourceReader is closed!");
            BinaryFormatter bf = new BinaryFormatter(null, new StreamingContext(StreamingContextStates.File |
#if !FEATURE_PAL
            _typeLimitingBinder = new TypeLimitingDeserializationBinder();
            bf.Binder = _typeLimitingBinder;
#endif
            _objFormatter = bf;
            try {
                // Read ResourceManager header
                // Check for magic number
                int magicNum = _store.ReadInt32();
                if
                //
                //
                //
                int
                if

                } e
                    // Read in type name for a suitable ResourceReader
```

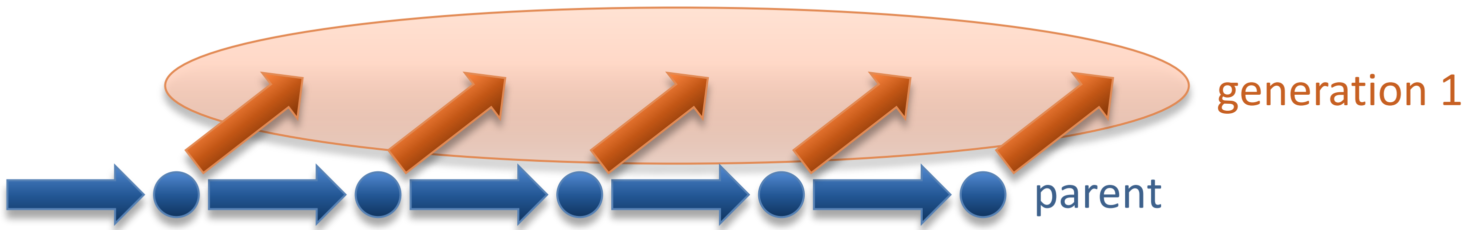
```
public virtual int ReadInt32() {
    if (m_isMemoryStream) {
        // read directly from MemoryStream buffer
        MemoryStream mStream = m_stream as MemoryStream;
        BCLDebug.Assert(mStream != null, "m_stream as MemoryStream != null");

        return mStream.InternalReadInt32();
    }
    else
    {
        FillBuffer(4);
        return (int)(m_buffer[0] | m_buffer[1] << 8 | m_buffer[2] << 16 | m_buffer[3] << 24);
    }
}
```

# Pex – Test Input Generation



# SAGE

- Apply DART to large applications (not units).
- Start with well-formed input (not random).
- Combine with generational search (not DFS).
  - Negate 1-by-1 each constraint in a path constraint.
  - Generate many children for each parent run.




Microsoft Research

# Zero to Crash in 10 Generations

- Starting with 100 zero bytes …
- SAGE generates a crashing test for Media1 parser

```
00000000h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000010h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000020h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000030h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000040h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000050h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000060h: 00 00 00 00                                     ; ....
```

Generation 0 – seed file

Microsoft Research

# Zero to Crash in 10 Generations

- Starting with 100 zero bytes …
- SAGE generates a crashing test for Media1 parser

```
00000000h: 52 49 46 46 00 00 00 00 00 00 00 00 00 00 00 00 ; RIFF............
00000010h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000020h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000030h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000040h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000050h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000060h: 00 00 00 00                                     ; ....
```

Generation 1

# Zero to Crash in 10 Generations

- Starting with 100 zero bytes …
- SAGE generates a crashing test for Media1 parser

```
00000000h: 52 49 46 46 00 00 00 00 ** ** ** 20 00 00 00 00 ; RIFF....*** ....
00000010h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000020h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000030h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000040h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000050h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000060h: 00 00 00 00                                     ; ....
```

Generation 2

Microsoft Research

# Zero to Crash in 10 Generations

- Starting with 100 zero bytes …
- SAGE generates a crashing test for Media1 parser

```
00000000h: 52 49 46 46 3D 00 00 00 ** ** ** 20 00 00 00 00 ; RIFF=...*** ....
00000010h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000020h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000030h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000040h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000050h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000060h: 00 00 00 00                                     ; ....
```

Generation 3

Microsoft Research

# Zero to Crash in 10 Generations

- Starting with 100 zero bytes …
- SAGE generates a crashing test for Media1 parser

```
00000000h: 52 49 46 46 3D 00 00 00 ** ** ** 20 00 00 00 00 ; RIFF=...*** ....
00000010h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000020h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000030h: 00 00 00 00 73 74 72 68 00 00 00 00 00 00 00 00 ; ....strh........
00000040h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000050h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000060h: 00 00 00 00                                     ; ....
```

Generation 4

Microsoft Research

# Zero to Crash in 10 Generations

- Starting with 100 zero bytes …
- SAGE generates a crashing test for Media1 parser

```
00000000h: 52 49 46 46 3D 00 00 00 ** ** ** 20 00 00 00 00 ; RIFF=...*** ....
00000010h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000020h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000030h: 00 00 00 00 73 74 72 68 00 00 00 00 76 69 64 73 ; ....strh....vids
00000040h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000050h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000060h: 00 00 00 00                                     ; ....
```

Generation 5

# Zero to Crash in 10 Generations

- Starting with 100 zero bytes …
- SAGE generates a crashing test for Media1 parser

```
00000000h: 52 49 46 46 3D 00 00 00 ** ** ** 20 00 00 00 00 ; RIFF=...*** ....
00000010h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000020h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000030h: 00 00 00 00 73 74 72 68 00 00 00 00 76 69 64 73 ; ....strh....vids
00000040h: 00 00 00 00 73 74 72 66 00 00 00 00 00 00 00 00 ; ....strf........
00000050h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000060h: 00 00 00 00                                     ; ....
```

Generation 6

# Zero to Crash in 10 Generations

- Starting with 100 zero bytes …
- SAGE generates a crashing test for Media1 parser

```
00000000h: 52 49 46 46 3D 00 00 00 ** ** ** 20 00 00 00 00 ; RIFF=...***  ....
00000010h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000020h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000030h: 00 00 00 00 73 74 72 68 00 00 00 00 76 69 64 73 ; ....strh....vids
00000040h: 00 00 00 00 73 74 72 66 00 00 00 00 28 00 00 00 ; ....strf....(...
00000050h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000060h: 00 00 00 00                                     ; ....
```

Generation 7

Microsoft Research

# Zero to Crash in 10 Generations

- Starting with 100 zero bytes …
- SAGE generates a crashing test for Media1 parser

```
00000000h: 52 49 46 46 3D 00 00 00 ** ** ** 20 00 00 00 00 ; RIFF=...*** ....
00000010h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000020h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000030h: 00 00 00 00 73 74 72 68 00 00 00 00 76 69 64 73 ; ....strh....vids
00000040h: 00 00 00 00 73 74 72 66 00 00 00 00 28 00 00 00 ; ....strf....(...
00000050h: 00 00 00 00 00 00 00 00 00 00 00 00 C9 9D E4 4E ; ............ÉäN
00000060h: 00 00 00 00                                     ; ....
```

Generation 8

Microsoft Research

# Zero to Crash in 10 Generations

- Starting with 100 zero bytes …
- SAGE generates a crashing test for Media1 parser

```
00000000h: 52 49 46 46 3D 00 00 00 ** ** ** 20 00 00 00 00 ; RIFF=...*** ....
00000010h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000020h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000030h: 00 00 00 00 73 74 72 68 00 00 00 00 76 69 64 73 ; ....strh....vids
00000040h: 00 00 00 00 73 74 72 66 00 00 00 00 28 00 00 00 ; ....strf....(...
00000050h: 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 ; ................
00000060h: 00 00 00 00                                     ; ....
```

Generation 9

Microsoft Research

# Zero to Crash in 10 Generations

- Starting with 100 zero bytes …
- SAGE generates a crashing test for Media1 parser

```
00000000h: 52 49 46 46 3D 00 00 00 ** ** ** 20 00 00 00 00 ; RIFF=...*** ....
00000010h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000020h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000030h: 00 00 00 00 73 74 72 68 00 00 00 00 76 69 64 73 ; ....strh....vids
00000040h: 00 00 00 00 73 74 72 66 B2 75 76 3A 28 00 00 00 ; ....strf²uv:(...
00000050h: 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 ; ................
00000060h: 00 00 00 00                                     ; ....
```

Generation 10 – CRASH

Microsoft Research

# SAGE (cont.)

- SAGE is very effective at finding bugs.
- Works on large applications.
- Fully automated
- Easy to deploy (x86 analysis – any language)
- Used in various groups inside Microsoft
- Powered by Z3.

# SAGE $\leftrightarrow$ Z3

- Formulas are usually big conjunctions.
- SAGE uses only the bitvector and array theories.
- Pre-processing step has a huge performance impact.
  - Eliminate variables.
  - Simplify formulas.
- Early unsat detection.

# Application Scenarios

"Big" and hard formulas

Thousands of "small" and easy formulas

Short timeout (< 5secs)

# Application Scenarios

"Big" and hard formulas

The Spec# Programming System

HAVOC

VCC

Thousands of "small" and easy formulas

Yogi

SLAM

Short timeout (< 5secs)

SAGE

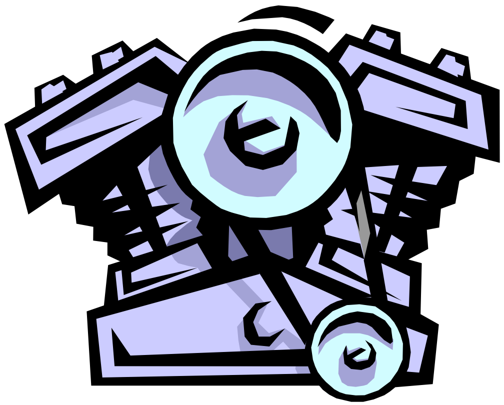
Microsoft Research

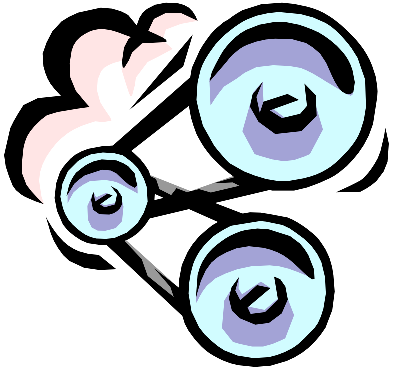
# Combining Engines

Current SMT solvers provide

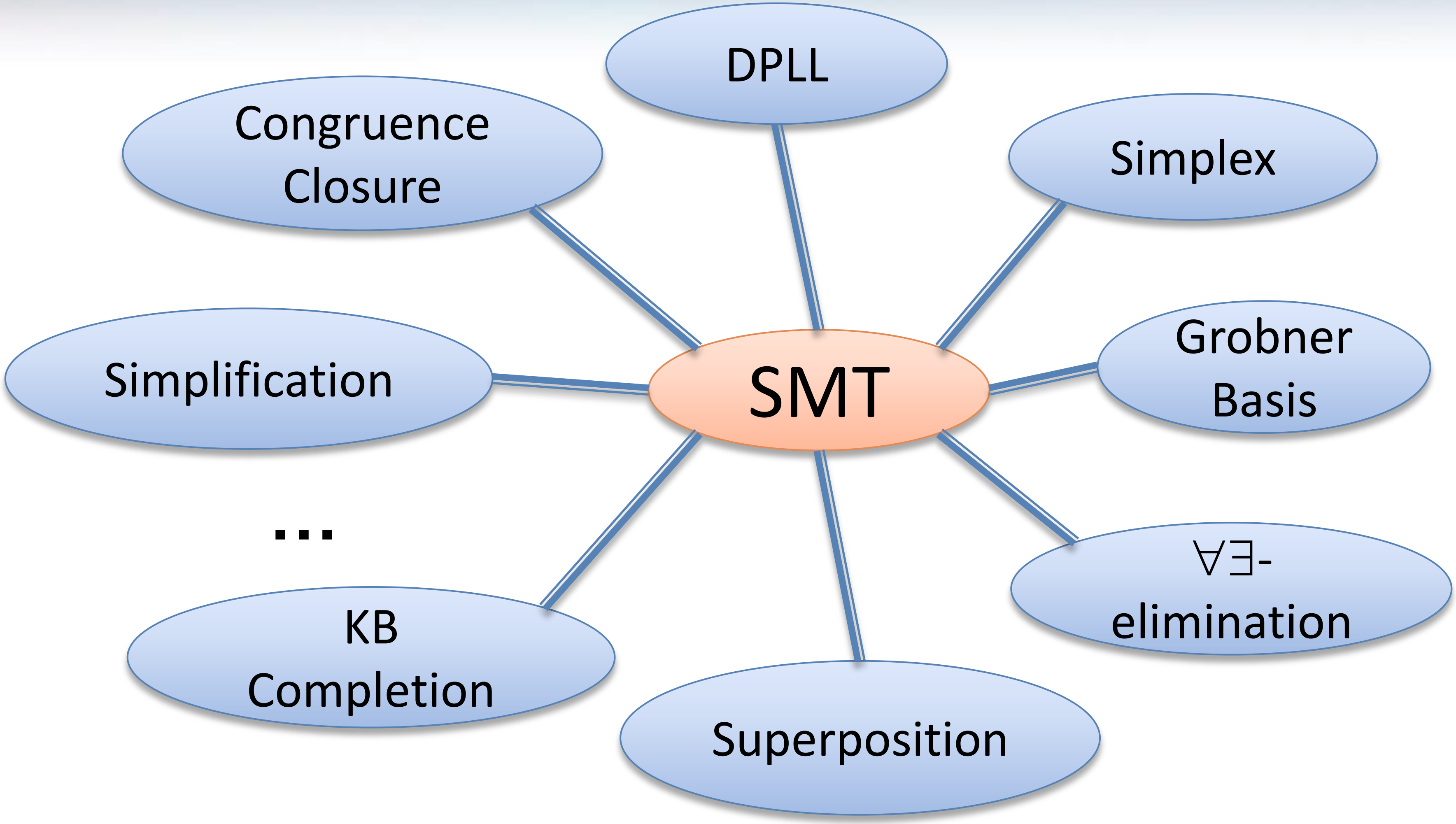
a combination

of different engines

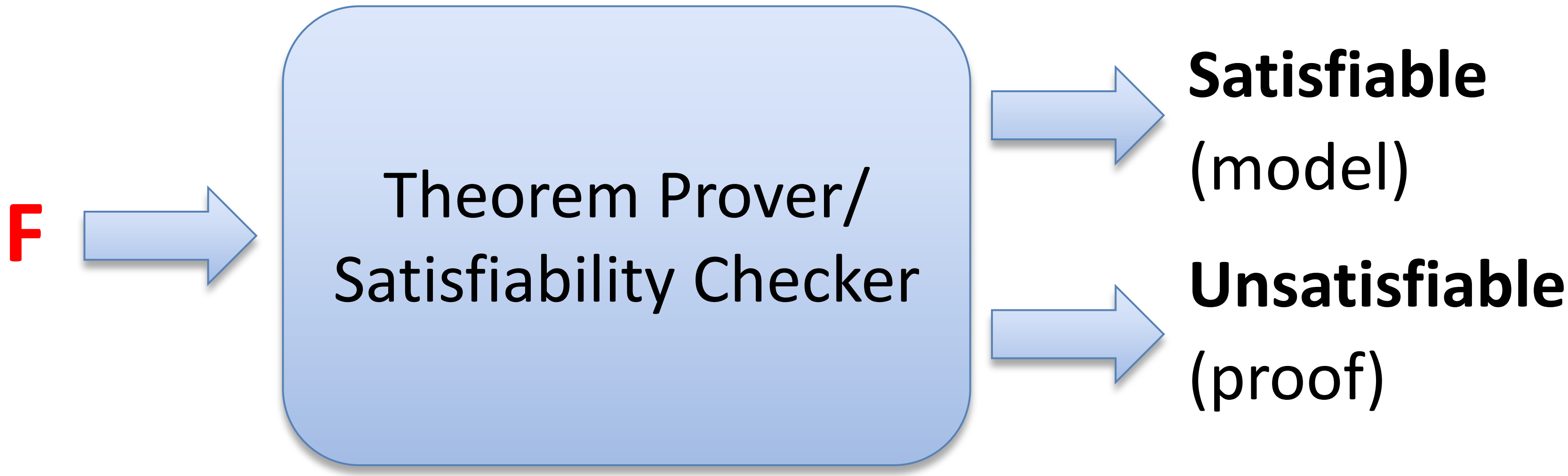




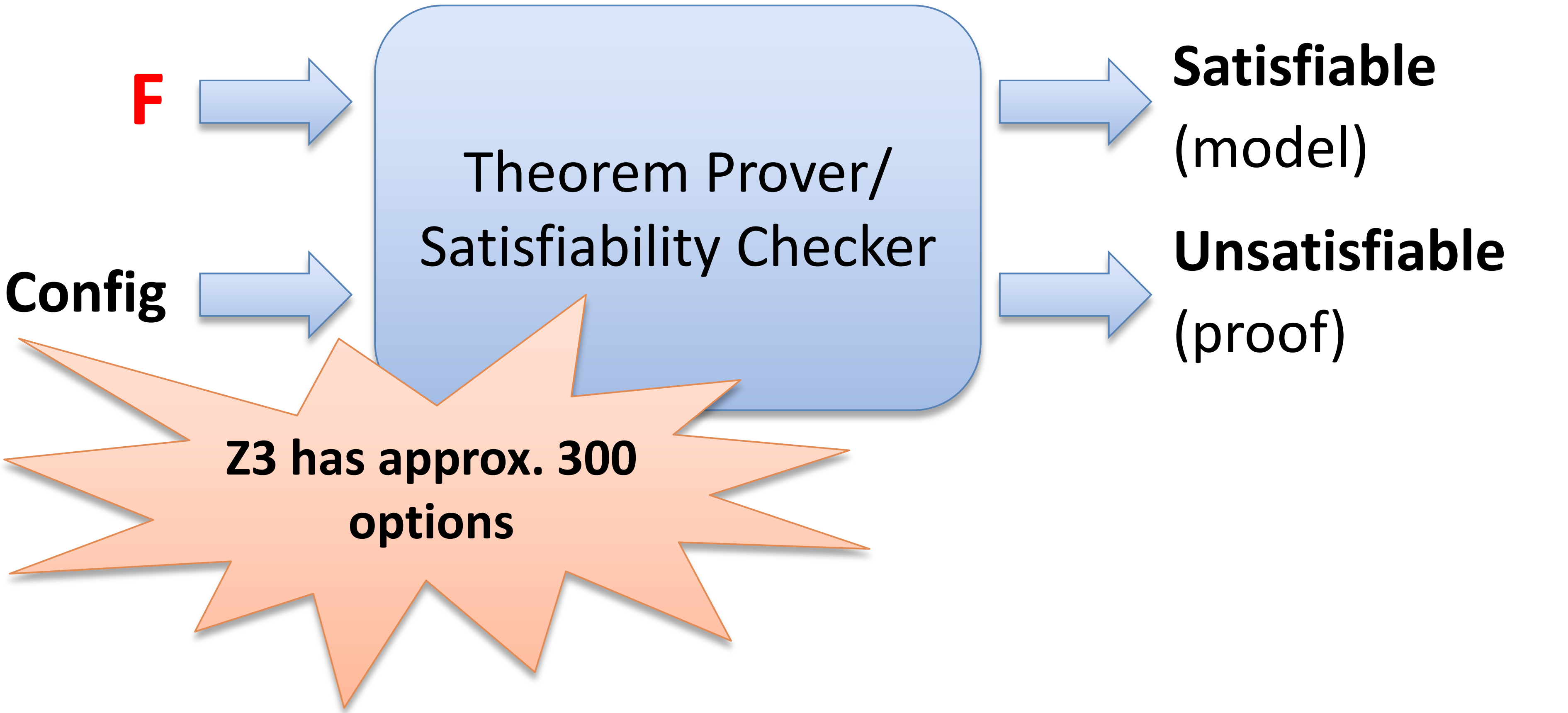
# Combining Engines



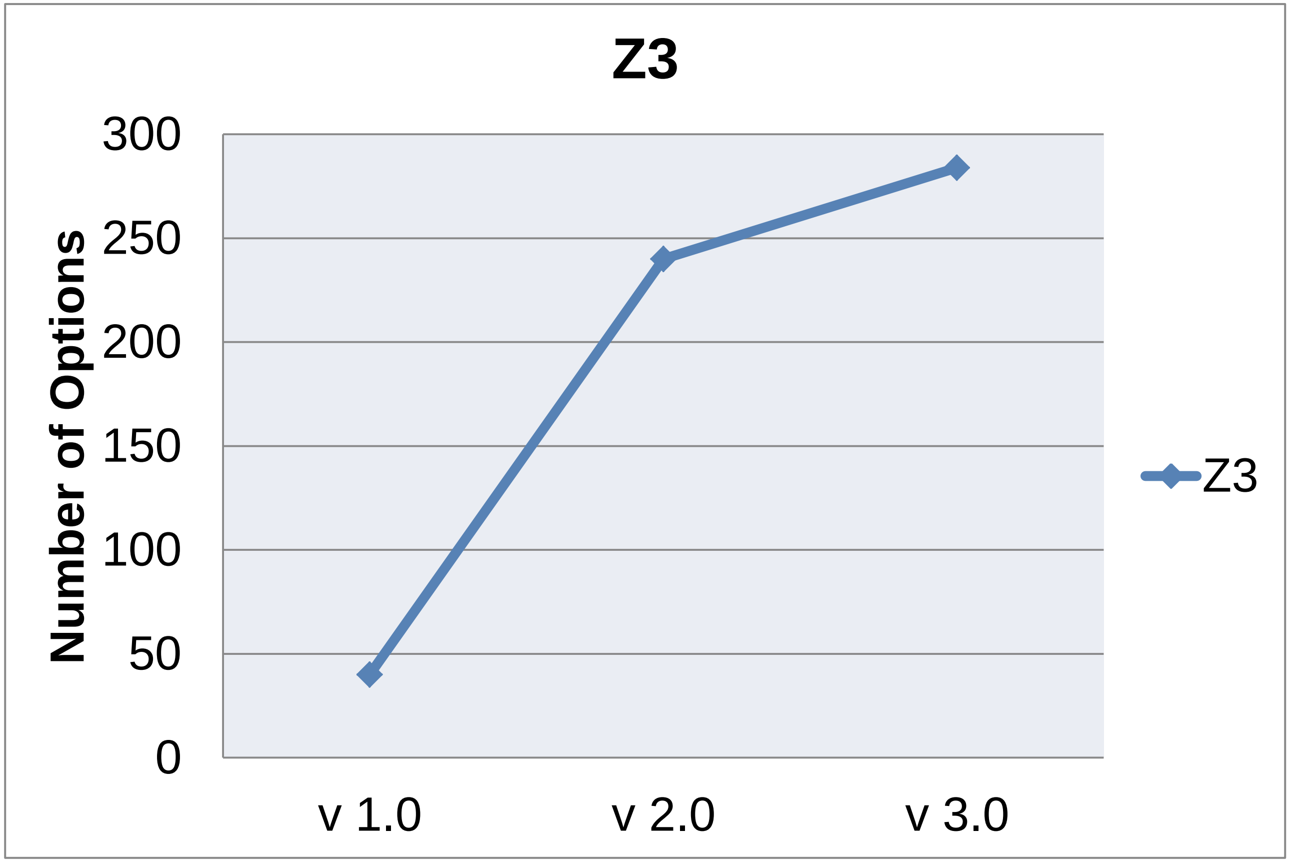
# Theorem Provers & Satisfiability Checkers



# Theorem Provers & Satisfiability Checkers



# Z3 number of options evolution



# Opening the “Black Box”

Actual feedback provided by Z3 users:

“Could you send me your CNF converter?”

“I want to implement my own search strategy.”

“I want to include these rewriting rules in Z3.”

“I want to apply a substitution to term *t*.”

“I want to compute the set of implied equalities.”

# The Strategy Challenge

To build theoretical and practical tools allowing users to exert strategic control over core heuristic aspects of high performance SMT solvers.

# What is a strategy?

Theorem proving as an exercise of combinatorial search

Strategies are adaptations of general search mechanisms which reduce the search space by tailoring its exploration to a particular class of formulas.

# The Need for "Strategies"

Different Strategies for Different Domains.

# The Need for "Strategies"

Different Strategies for Different Domains.

From timeout to 0.05 secs…

# Example in Quantified Bit-Vector Logic (QBVF)

Join work with C. Wintersteiger and Y. Hamadi
FMCAD 2010

QBVF = Quantifiers + Bit-vectors + uninterpreted functions

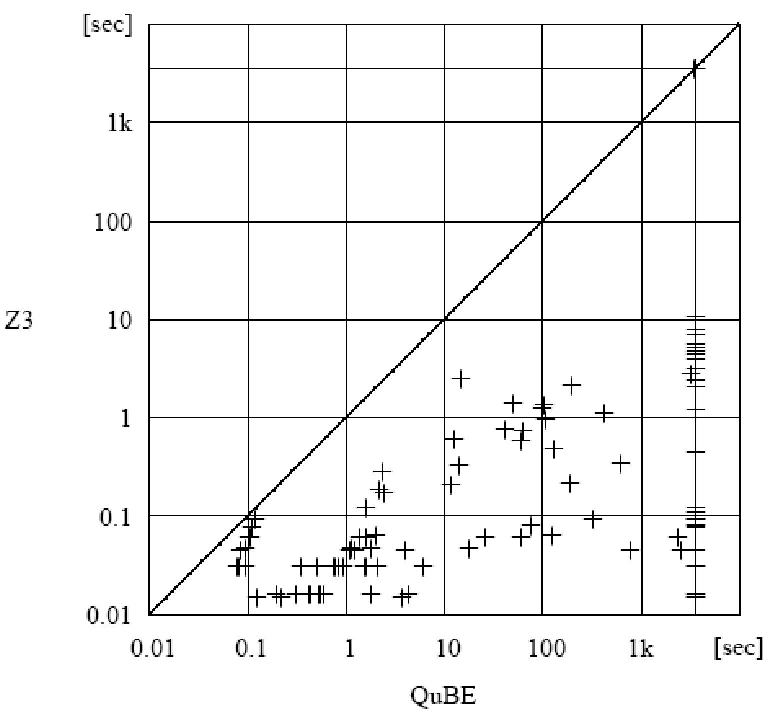
Hardware Fixpoint Checks.

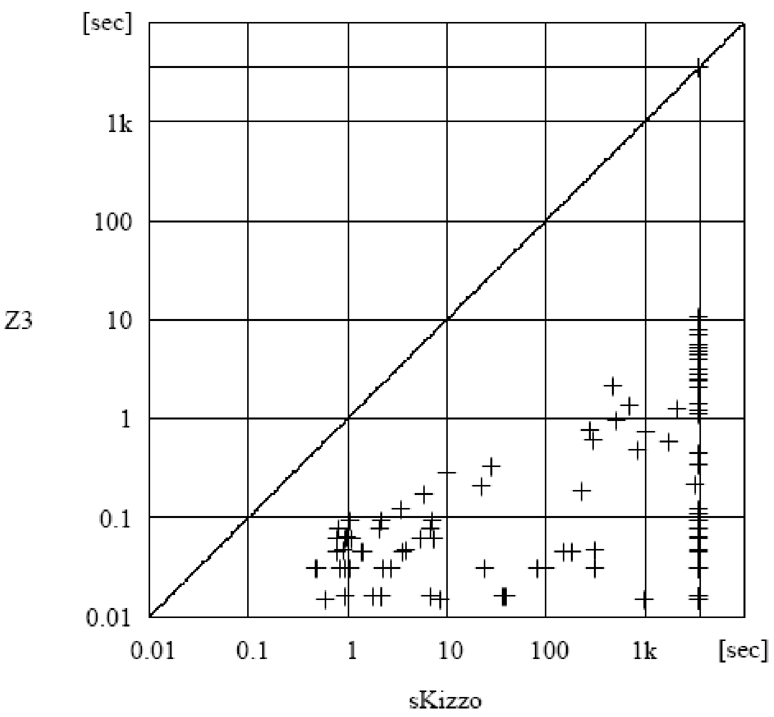
Given: $I[x]$ and $T[x, x']$

$$\forall x, x' \ . \ I[x] \wedge T^k[x, x'] \rightarrow \exists y, y' \ . I[y] \wedge T^{k-1}[y, y']$$

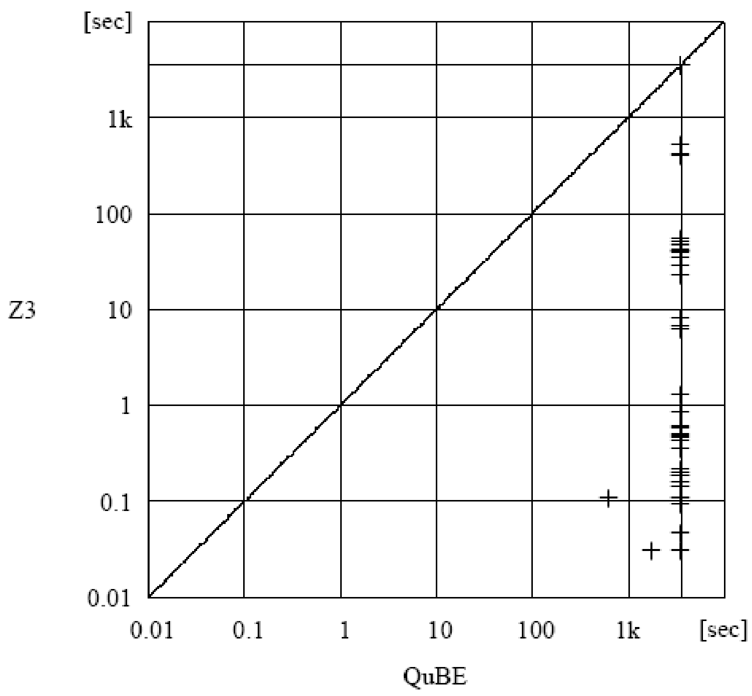
Ranking function synthesis.

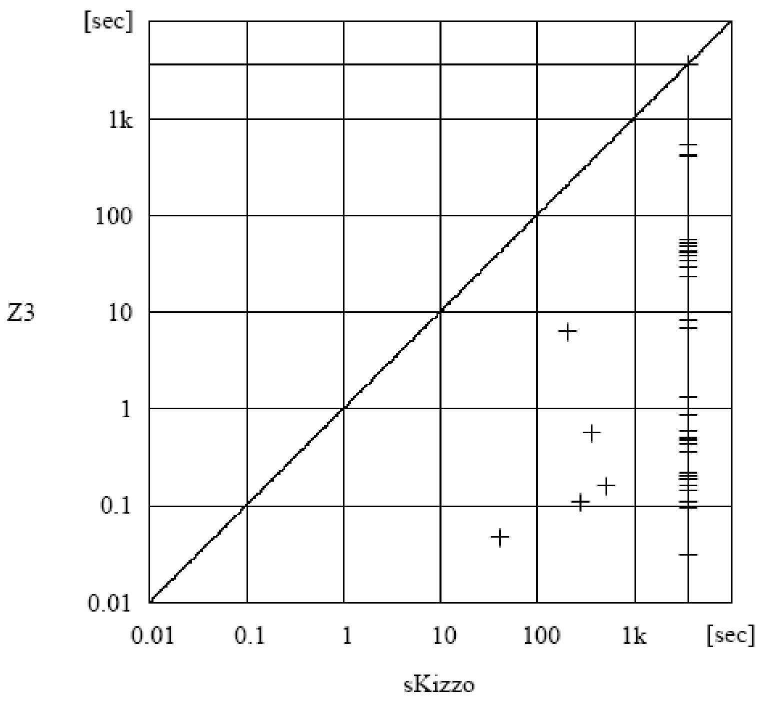
# Hardware Fixpoint Checks



# Ranking Function Synthesis



# Why is Z3 so fast in these benchmarks?

Z3 is using different engines:

rewriting, simplification, model checking, SAT, …

Z3 is using a customized **strategy**.

We could do it because

we have access to the source code.

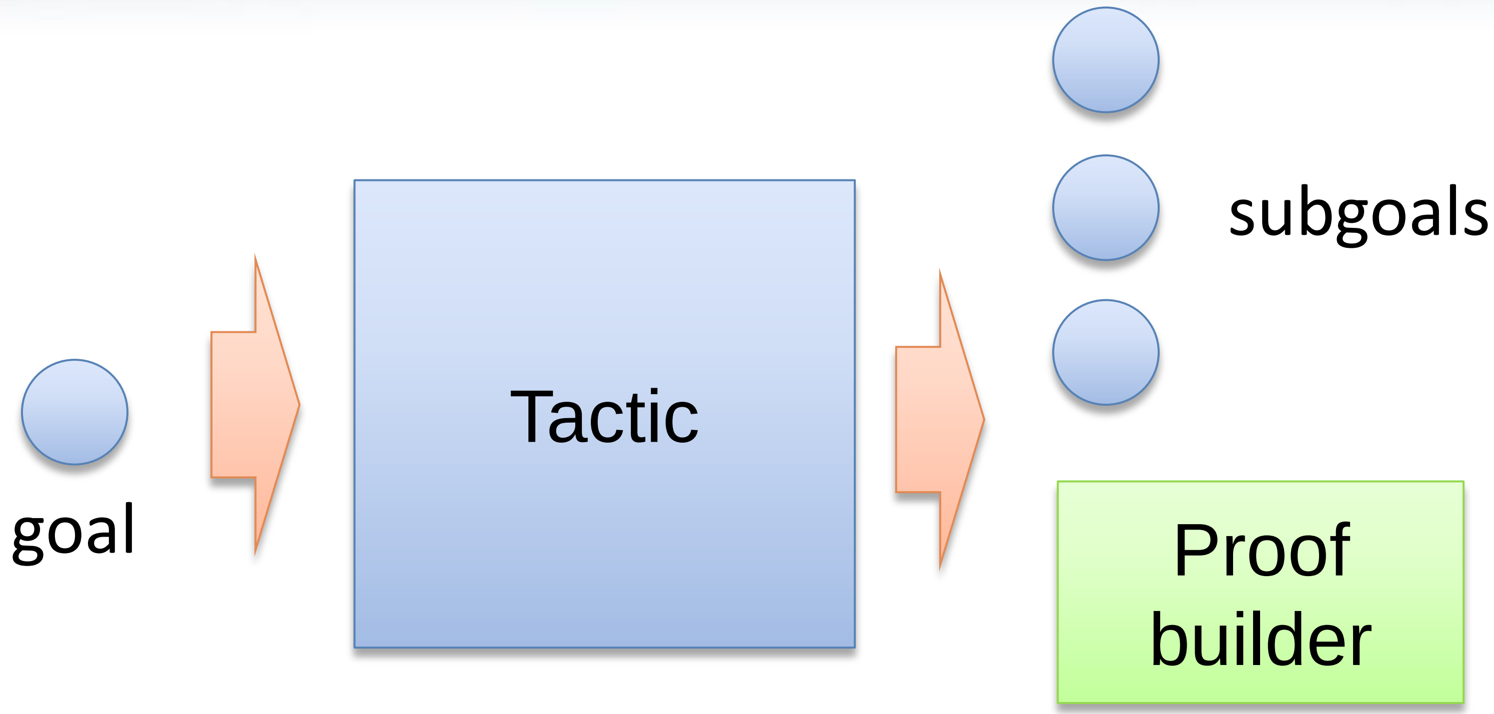
# The "Message"

SMT solvers are collections of little engines.

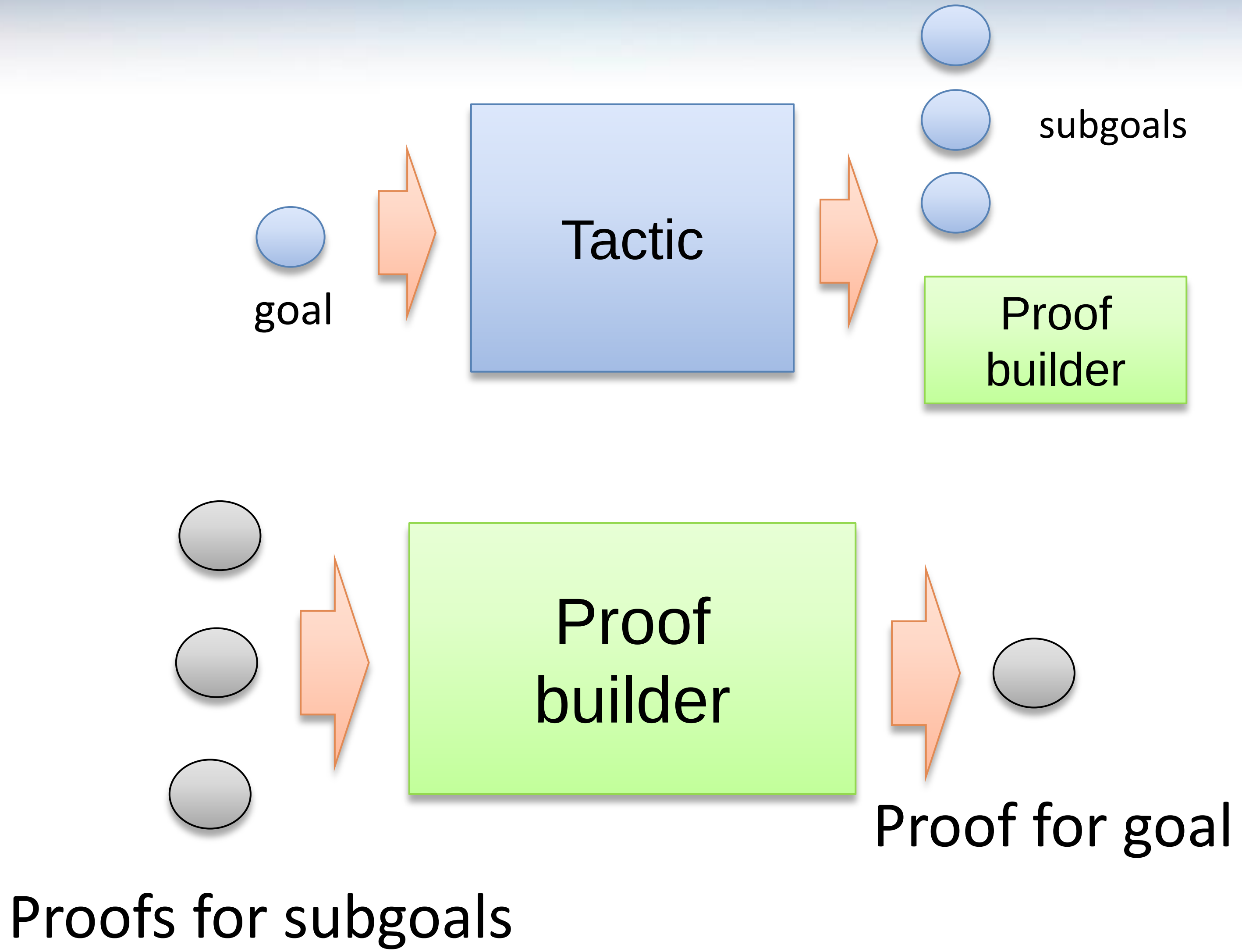
They should provide access to these engines.

Users should be able to define their own strategies.

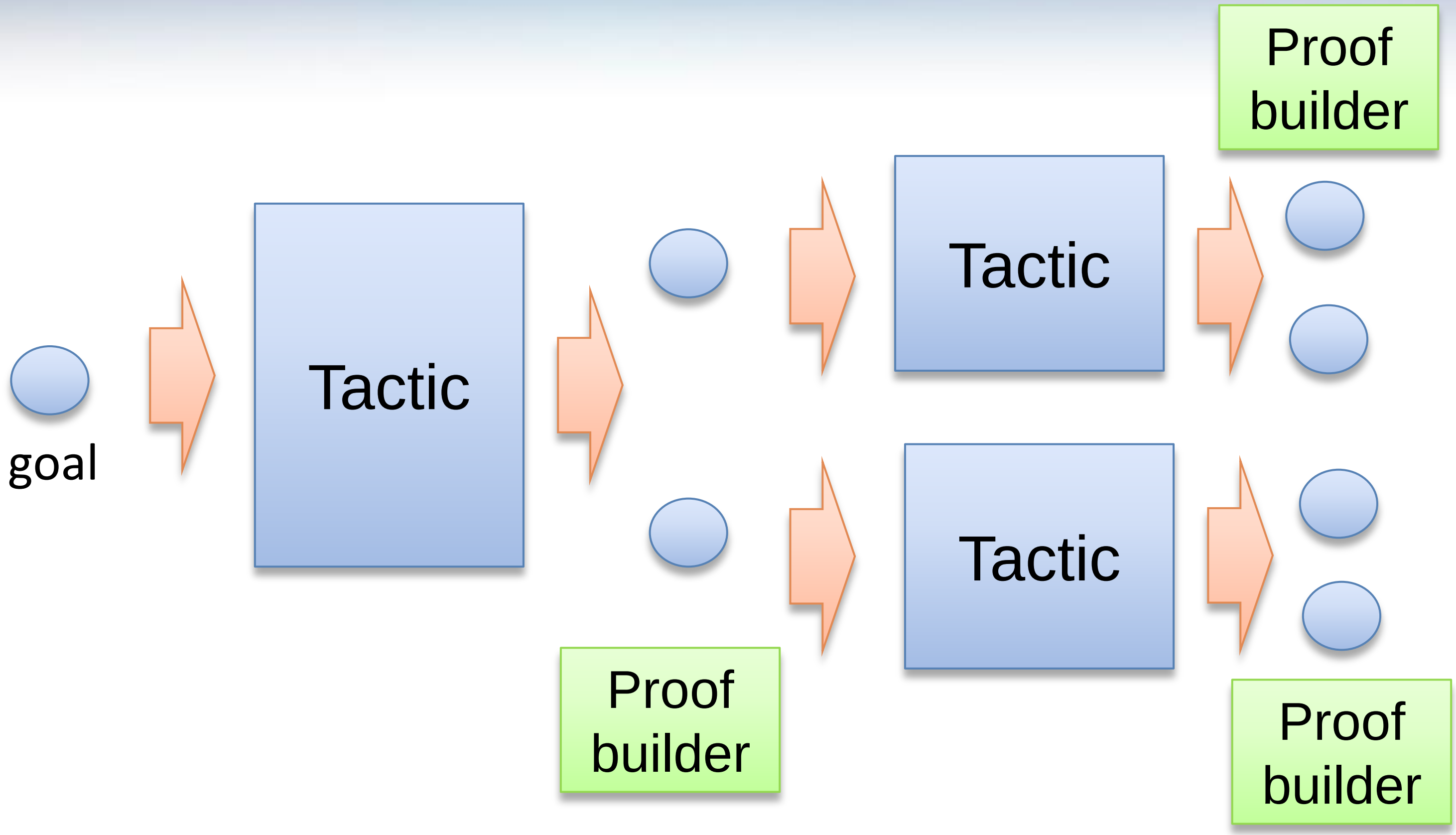
# Main inspiration: LCF-approach



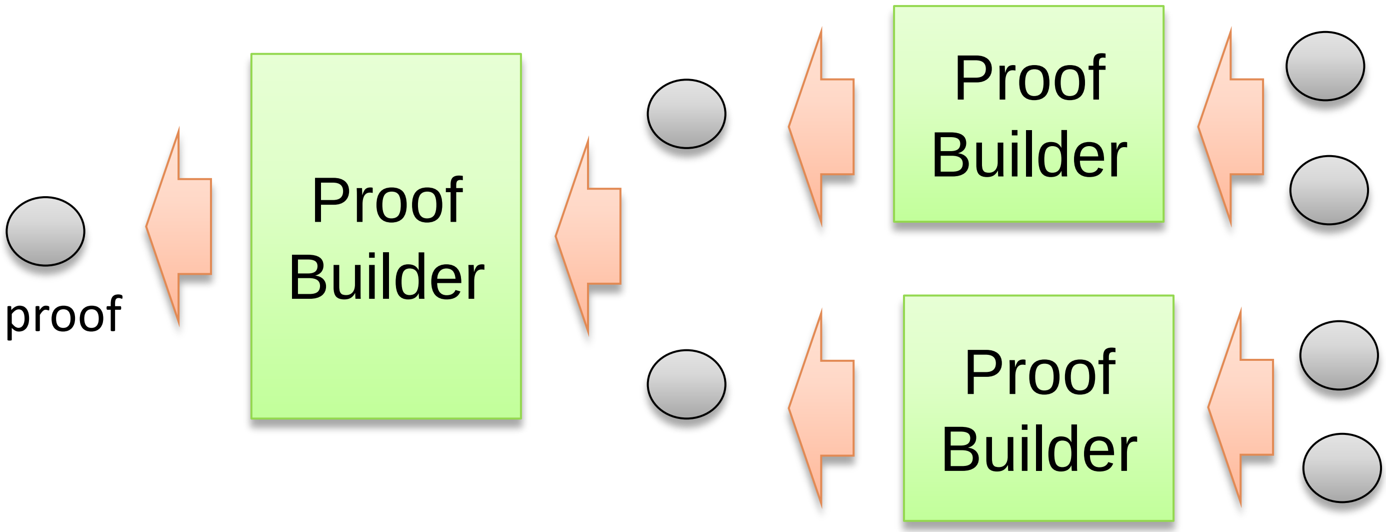
# Main inspiration: LCF-approach



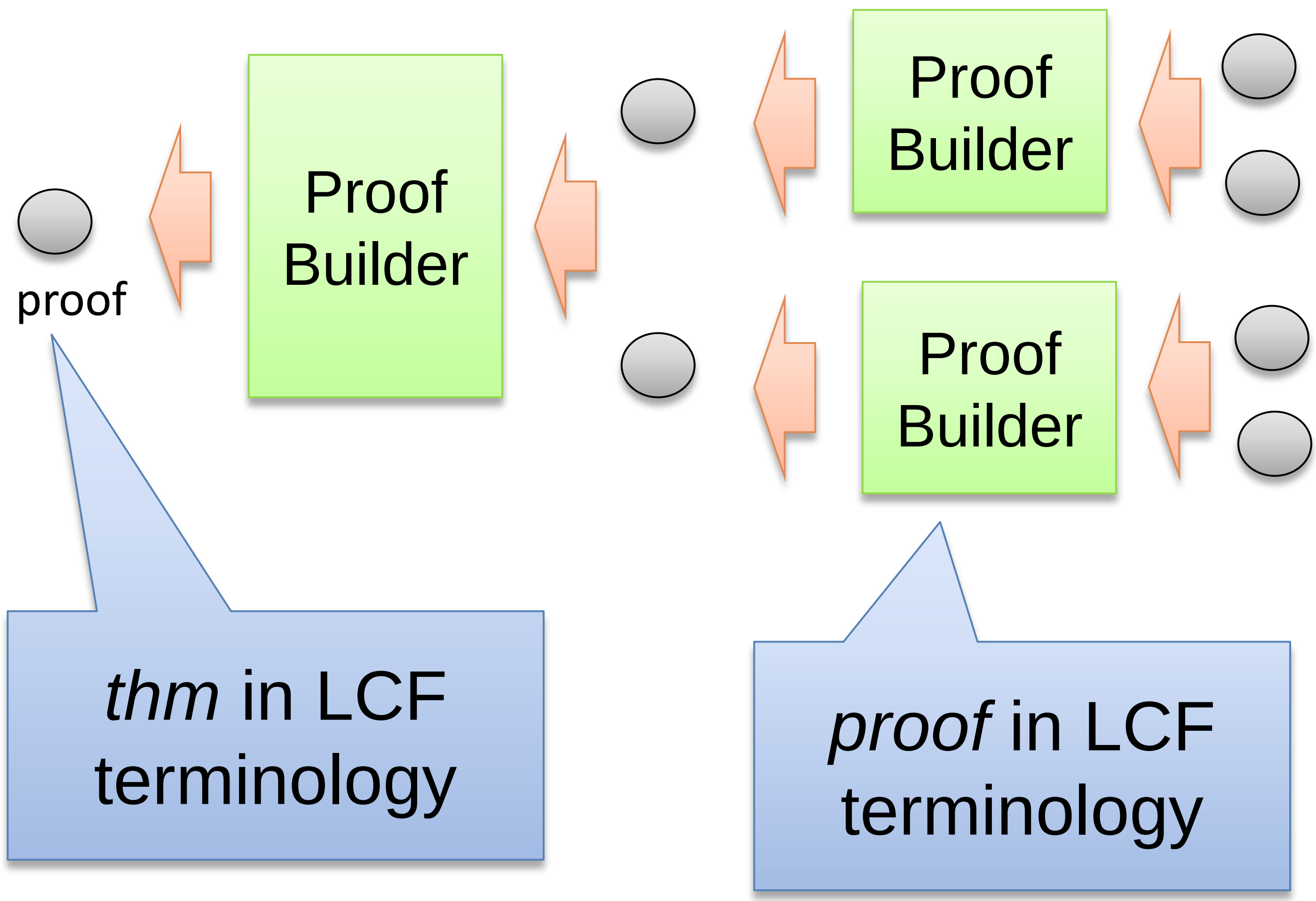
# Main inspiration: LCF-approach



# Main inspiration: LCF-approach



# Main inspiration: LCF-approach



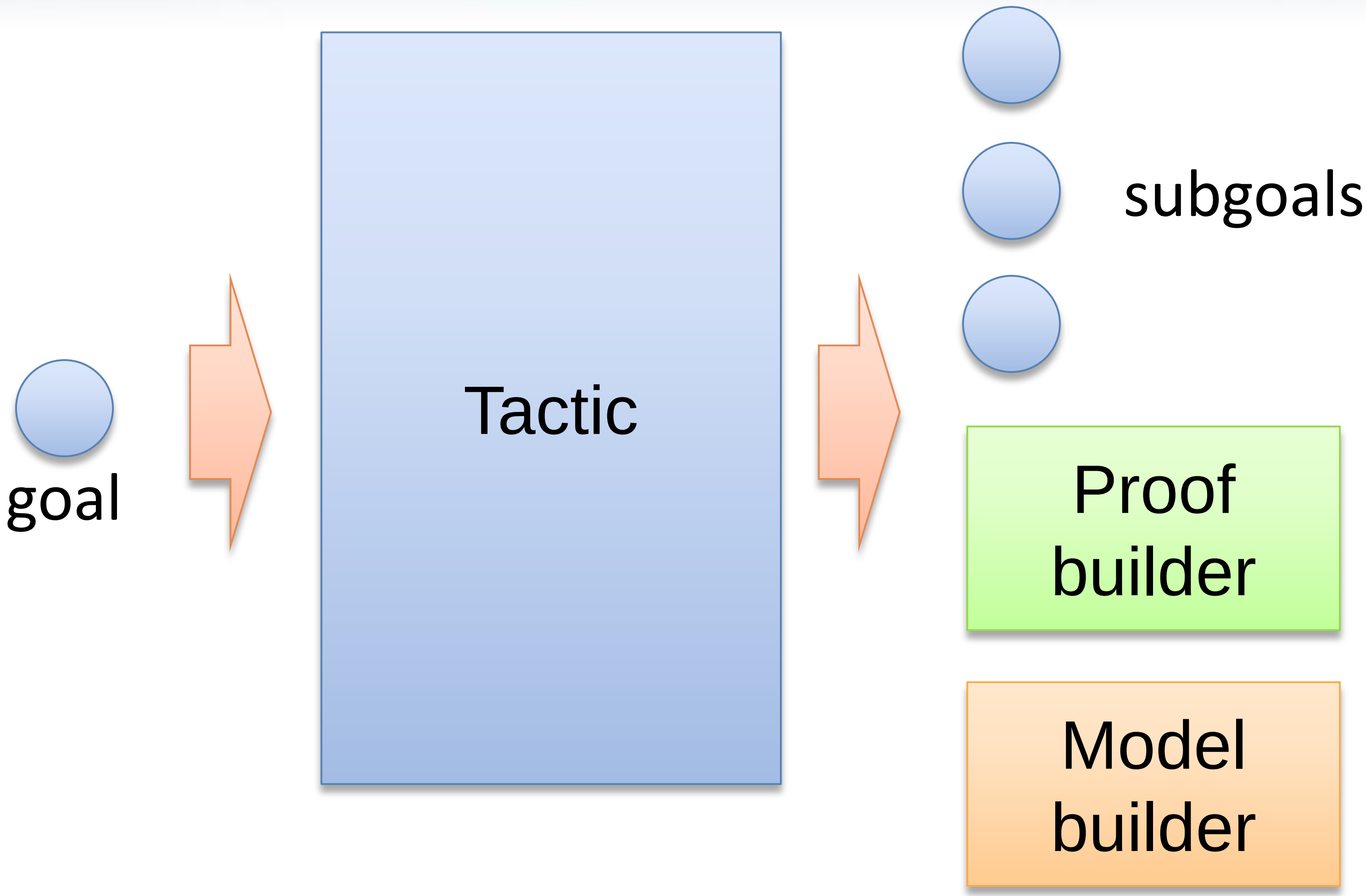
# Tacticals aka Combinators

then( Tactic , Tactic ) = Tactic

orelse( Tactic , Tactic ) = Tactic

repeat( Tactic ) = Tactic

# SMT Tactic



# SMT Tactic

$$
\begin{aligned}
\mathit{goal} &= \mathit{formula\ sequence} \times \mathit{attribute\ sequence} \\
\\
\mathit{proofconv} &= \mathit{proof\ sequence} \longrightarrow \mathit{proof} \\
\mathit{modelconv} &= \mathit{model} \times \mathit{nat} \longrightarrow \mathit{model} \\
\mathit{trt} &= \mathsf{sat}\ \mathit{model} \\
&\mid\ \ \mathsf{unsat}\ \mathit{proof} \\
&\mid\ \ \mathsf{unknown}\ \mathit{goal\ sequence} \times \mathit{modelconv} \times \mathit{proofconv} \\
&\mid\ \ \mathsf{fail} \\
\mathit{tactic} &= \mathit{goal} \longrightarrow \mathit{trt}
\end{aligned}
$$

# SMT Tactic

$$goal = formula\ sequence \times attribute\ sequence$$

$$\begin{aligned}
proofconv &= proof\ sequence \rightarrow proof \\
modelconv &= model \times nat \rightarrow model \\
trt &= \mathsf{sat}\ model \\
&\mid\ \ \mathsf{unsat}\ proof \\
&\mid\ \ \mathsf{unknown}\ goal\ sequence \times modelconv \times proofconv \\
&\mid\ \ \mathsf{fail} \\
tactic &= goal \rightarrow trt
\end{aligned}$$

end-game tactics:
never return unknown(sb, mc, pc)

# SMT Tactic

$$
\begin{aligned}
goal &= formula\ sequence \times attribute\ sequence \\
\\
proofconv &= proof\ sequence \rightarrow proof \\
modelconv &= model \times nat \rightarrow model \\
trt &= \mathsf{sat}\ model \\
&\mid\ \mathsf{unsat}\ proof \\
&\mid\ \mathsf{unknown}\ goal\ sequence \times modelconv \times proofconv \\
&\mid\ \mathsf{fail} \\
tactic &= goal \rightarrow trt
\end{aligned}
$$

non-branching tactics:
sb is a sigleton in
unknown(sb, mc, pc)

# Trivial goals

Empty goal [ ] is trivially satisfiable

False goal [ ..., false, ...] is trivially unsatisfiable

basic : tactic

# SMT Tactic example

$$[\ a = b + 1,\ (a < 0 \lor a > 0),\ b > 3\ ]$$

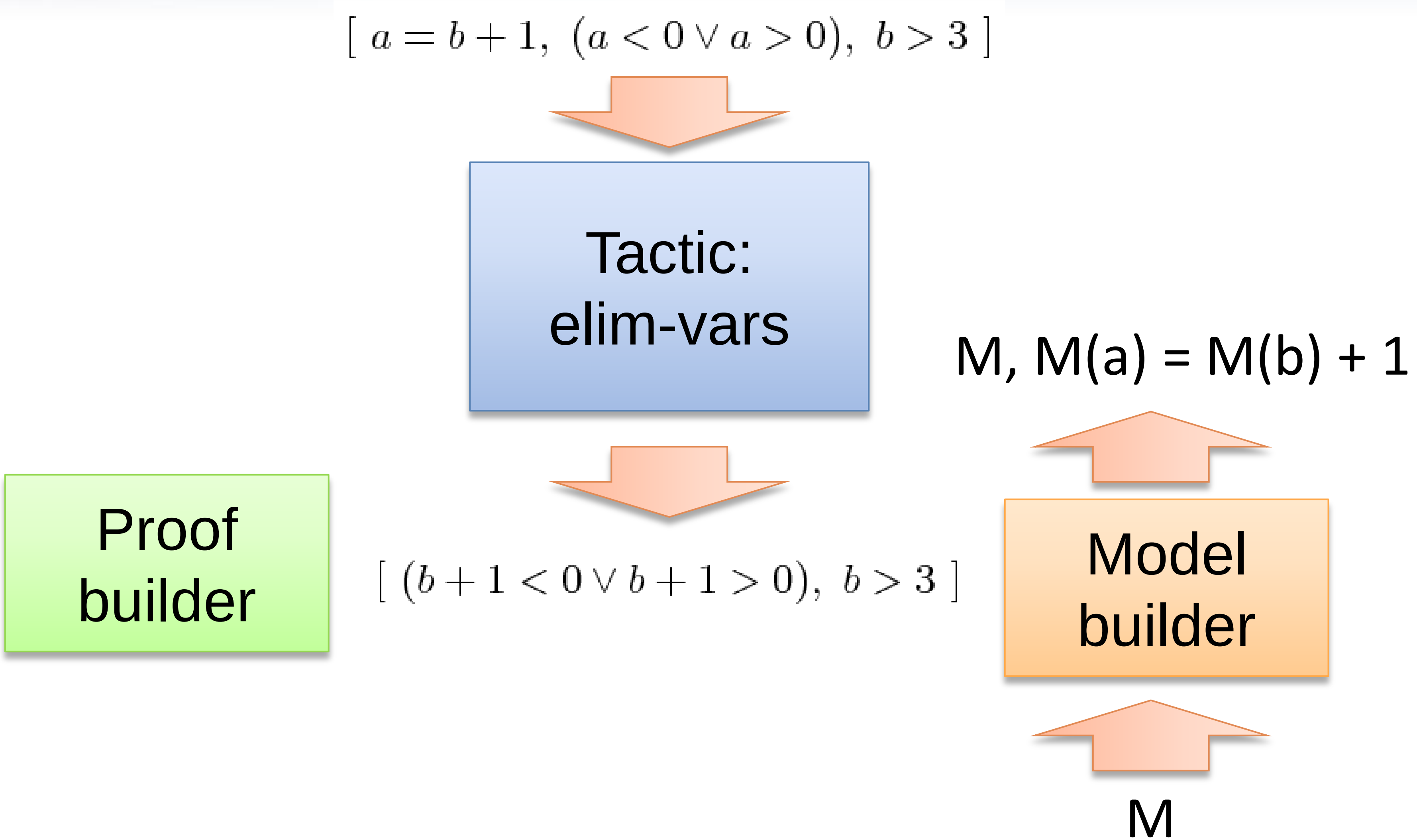
Tactic:
elim-vars

Proof
builder

$$[\ (b + 1 < 0 \lor b + 1 > 0),\ b > 3\ ]$$

Model
builder

# SMT Tactic example



# SMT Tactic example

$$[\ a = b + 1,\ \ (a < 0 \lor a > 0),\ \ b > 3\ ]$$

Tactic:
split-or

Proof
builder

$$[\ a = b + 1,\ \ a < 0,\ \ b > 3\ ]$$

$$[\ a = b + 1,\ \ a > 0,\ \ b > 3\ ]$$

Model
builder

# SMT Tactics

simplify
nnf
cnf
tseitin
lift-if
bitblast
gb
vts

propagate-bounds
propagate-values
split-ineqs
split-eqs
rewrite
p-cad
sat
solve-eqs

# SMT Tacticals

then : $(tactic \times tactic) \longrightarrow tactic$

then$(t_1, t_2)$ applies $t_1$ to the given goal and $t_2$ to every subgoal produced by $t_1$.

then$*$ : $(tactic \times tactic\ sequence) \longrightarrow tactic$

then$*(t_1, [t_{2_1}, ..., t_{2_n}])$ applies $t_1$ to the given goal, producing subgoals $g_1, ..., g_m$. If $n \neq m$, the tactic fails. Otherwise, it applies $t_{2_i}$ to every goal $g_i$.

orelse : $(tactic \times tactic) \longrightarrow tactic$

orelse$(t_1, t_2)$ first applies $t_1$ to the given goal, if it fails then returns the result of $t_2$ applied to the given goal.

par : $(tactic \times tactic) \longrightarrow tactic$

par$(t_1, t_2)$ excutes $t_1$ and $t_2$ in parallel.

# SMT Tacticals

$$\mathsf{then}(\mathsf{skip}, t) = \mathsf{then}(t, \mathsf{skip}) = t$$

$$\mathsf{orelse}(\mathsf{fail}, t) = \mathsf{orelse}(t, \mathsf{fail}) = t$$

# SMT Tacticals

repeat : $tactic \longrightarrow tactic$

Keep applying the given tactic until no subgoal is modified by it.

repeatupto : $tactic \times nat \longrightarrow tactic$

Keep applying the given tactic until no subgoal is modified by it, or the maximum number of iterations is reached.

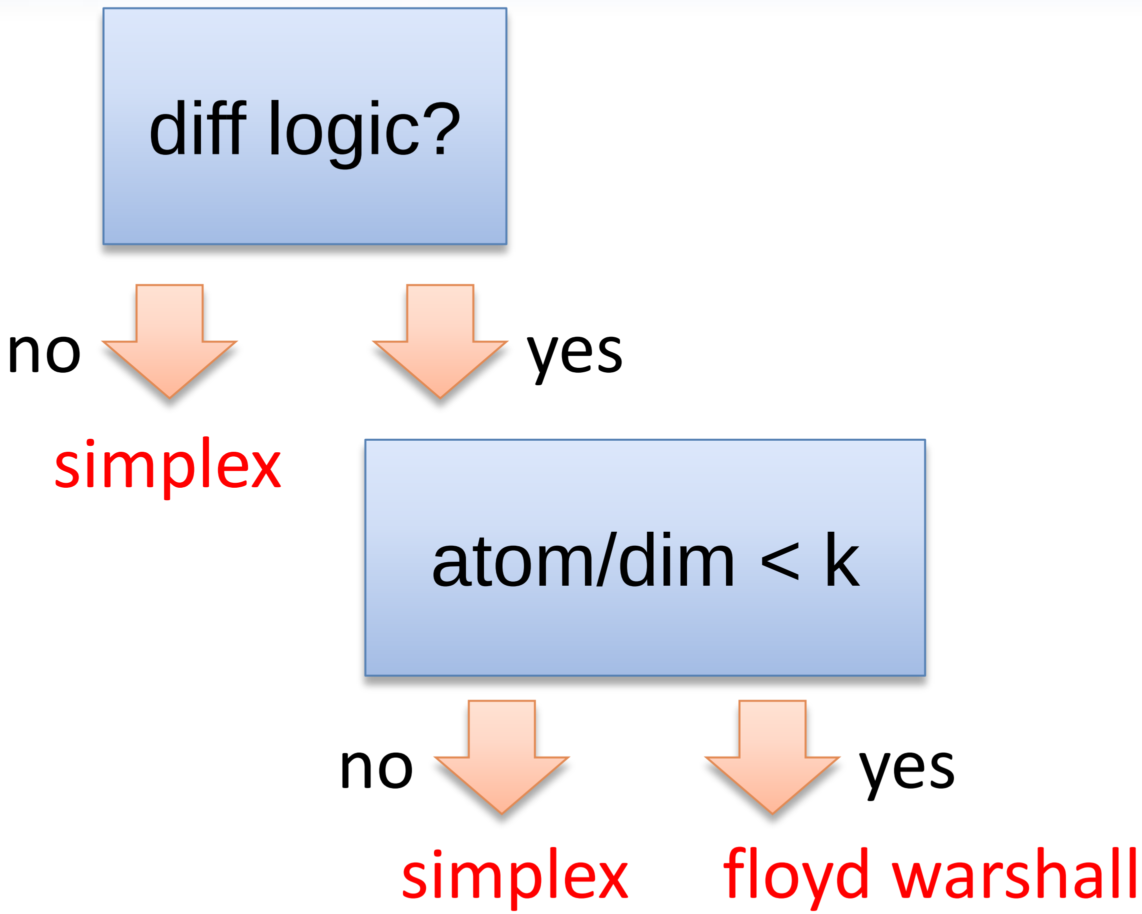
tryfor : $tactic \times seconds \longrightarrow tactic$

tryfor$(t, k)$ returns the value computed by tactic $t$ applied to the given goal if this value is computed within $k$ seconds, otherwise it fails.

# Feature / Measures

Probing structural features of formulas.

# Feature / Measures: Yices Strategy



# Feature / Measures: Yices Strategy

$$\text{orelse}(\text{then}(\text{failif}(\text{diff} \wedge \frac{\text{atom}}{\text{dim}} > \text{k}),\ \text{simplex}),\ \text{floydwarshall})$$

Fail if condition is not satisfied.
Otherwise, do nothing.

# Feature / Measures: Examples

bw: Sum total bit-width of all rational coefficients of polynomials in case.
diff: True if the formula is in the difference logic fragment.
linear: True if all polynomials are linear.
dim: Number of arithmetic constants.
atoms: Number of atoms.
degree: Maximal total multivariate degree of polynomials.
size: Total formula size.

# Tacticals: syntax sugar

$$\mathsf{if}(c,\ t_1,\ t_2) = \mathsf{orelse}(\mathsf{then}(\mathsf{failif}(\neg c), t_1), t_2)$$

$$\mathsf{when}(c,\ t) = \mathsf{if}(c,\ t,\ \mathsf{skip})$$

# Under/Over-Approximations

Under-approximation

unsat answers cannot be trusted

Over-approximation

sat answers cannot be trusted

# Under/Over-Approximations

Under-approximation

model finders

Over-approximation

proof finders

# Under/Over-Approximations

Under-approximation

$S \rightarrow S \cup S'$

Over-approximation

$S \rightarrow S \setminus S'$

# Under/Over-Approximations

Under-approximation

Example: QF_NIA model finders

add bounds to unbounded variables (and blast)

Over-approximation

Example: Boolean abstraction

# Under/Over-Approximations

Combining under and over is bad!

sat and unsat answers cannot be trusted.

# Tracking: under/over-approximations

In principle, proof and model converters can check if the resultant models and proofs are valid.

# Tracking: under/over-approximations

In principle, proof and model converters can check if the resultant models and proofs are valid.

Problem: if it fails what do we do?

# Tracking: under/over-approximations

In principle, proof and model converters can check if the resultant models and proofs are valid.

Problem: if it fails what do we do?

We want to write tactics that can check whether a goal is the result of an abstraction or not.

# Tracking: under/over-approximations

Solution

Associate an precision attribute to each goal.

# Goal Attributes

Store extra logical information

Examples:

precision markers

goal depth

polynomial factorizations

# Decision Engines as Tacticals

AP-CAD ( tactic ) = tactic

# Decision Engines as Tactical

$$\mathsf{then}(\mathsf{preprocess},\ \mathsf{smt}(\mathsf{finalcheck}))$$

# Strategy: Example

then(then(simplify, gaussian), orelse(modelfinder, smt(apcad(icp))))

# RAHD Calculemus Strategy

| | dim | deg | div | calc-0 | calc-1 | calc-2 | qepcad-b | redlog/rlqe | redlog/rlcad |
|---|---|---|---|---|---|---|---|---|---|
| P0 | 5 | 4 | N | .91 | 1.59 | 1.7 | 416.45* | 40.4 | - |
| P1 | 6 | 4 | N | 1.69 | 3.08 | 3.42 | -* | - | - |
| P2 | 5 | 4 | N | 1.34 | 2.41 | 2.62 | -* | - | - |
| P3 | 5 | 4 | N | 1.52 | 2.56 | 2.75 | -* | - | - |
| P4 | 5 | 4 | N | 1.14 | 2.02 | 2.16 | -* | - | - |
| P5 | 14 | 2 | N | .25 | .26 | .27 | -* | 97.4 | - |
| P6 | 11 | 5 | N | 147.4 | .07 | .06 | -* | <.01 | <.01 |
| P7 | 8 | 2 | N | .05 | <.01 | <.01 | .08 | <.01 | <.01 |
| P8 | 7 | 32 | N | 4.5 | .1 | <.01 | 8.38 | <.01 | - |
| P9 | 7 | 16 | N | 4.51 | .15 | <.01 | .29 | .01 | 6.7 |
| P10 | 7 | 12 | N | 100.74 | 20.76 | 8.85 | -* | - | - |
| P11 | 6 | 2 | Y | 1.6 | .5 | .53 | .01 | .01 | .05 |
| P12 | 5 | 3 | N | .78 | .3 | .36 | .02 | .01 | .07 |
| P13 | 4 | 10 | N | 3.83 | 3.95 | 4.02 | -* | - | - |
| P14 | 2 | 2 | N | 4.55 | 1.67 | .07 | .01 | - | - |
| P15 | 4 | 3 | Y | .177 | .2 | .12 | .01 | <.01 | <.01 |
| P16 | 4 | 2 | N | 9.99 | 2.17 | 2.1 | .02 | <.01 | <.01 |
| P17 | 4 | 2 | N | .62 | .59 | .65 | .28 | .02 | .61 |
| P18 | 4 | 2 | N | 1.25 | 1.28 | 1.27 | .01 | <.01 | <.01 |
| P19 | 3 | 6 | Y | 3.34 | 1.72 | 2.08 | .02 | .01 | .7 |
| P20 | 3 | 4 | N | 1.18 | .65 | .65 | .01 | <.01 | .3 |
| P21 | 3 | 2 | N | .02 | .03 | <.01 | .02 | .01 | .1 |
| P22 | 2 | 4 | N | <.01 | <.01 | <.01 | .01 | <.01 | <.01 |
| P23 | 2 | 2 | Y | <.01 | <.01 | <.01 | <.01 | <.01 | <.01 |

# Z3 QF_LIA Strategy

then(preamble, orelse(mf, pb, bounded, smt))

Simplification
Constant propagation
Interval propagation
Contextual simplification
If-then-else elimination
Gaussian elimination
Unconstrained terms

# Challenge: small step configuration

proof procedure as a transition system

Abstract DPLL, DPLL(T), Abstract GB, cutsat, …

UnitPropagate :

$$M \parallel F, C \vee l \quad \Longrightarrow \quad M\, l \parallel F, C \vee l \quad \text{if} \begin{cases} M \models \neg C \\ l \text{ is undefined in } M \end{cases}$$

PureLiteral :

$$M \parallel F \quad \Longrightarrow \quad M\, l \parallel F \quad \text{if} \begin{cases} l \text{ occurs in some clause of } F \\ \neg l \text{ occurs in no clause of } F \\ l \text{ is undefined in } M \end{cases}$$

Decide :

$$M \parallel F \quad \Longrightarrow \quad M\, l^{\mathsf{d}} \parallel F \quad \text{if} \begin{cases} l \text{ or } \neg l \text{ occurs in a clause of } F \\ l \text{ is undefined in } M \end{cases}$$

Fail :

$$M \parallel F, C \quad \Longrightarrow \quad \mathit{FailState} \quad \text{if} \begin{cases} M \models \neg C \\ M \text{ contains no decision literals} \end{cases}$$

Backtrack :

$$M\, l^{\mathsf{d}}\, N \parallel F, C \quad \Longrightarrow \quad M\, \neg l \parallel F, C \quad \text{if} \begin{cases} M\, l^{\mathsf{d}}\, N \models \neg C \\ N \text{ contains no decision literals} \end{cases}$$

# Challenge: small step configuration

proof procedure as a transition system

Abstract DPLL, DPLL(T), Abstract GB, cutsat, …

**Challenge**:

Efficient strategic control

Decide :

$$M \parallel F \implies M\, l^{\mathsf{d}} \parallel F \quad \text{if} \begin{cases} l \text{ or } \neg l \text{ occurs in a clause of } F \\ l \text{ is undefined in } M \end{cases}$$

Fail :

$$M \parallel F, C \implies \mathit{FailState} \quad \text{if} \begin{cases} M \models \neg C \\ M \text{ contains no decision literals} \end{cases}$$

Backtrack :

$$M\, l^{\mathsf{d}}\, N \parallel F, C \implies M\, \neg l \parallel F, C \quad \text{if} \begin{cases} M\, l^{\mathsf{d}}\, N \models \neg C \\ N \text{ contains no decision literals} \end{cases}$$

# Conclusion

Different domains need different strategies.

We must expose the little engines in SMT solvers.

Interaction between different engines is a must.

Tactic and Tacticals: big step approach.

More transparency.