

# ICE: An Integrated Comprehension Environment

1

Dean Pucsek, Jonah Wall, Nieraj Singh,  
Jennifer Baldwin, Celina Gibbs & Yvonne Coady

University of Victoria

NSERC Workshop on  
Malware Analysis and  
Fingerprinting



# Why New Tools?

2

- Current tools aren't cutting it!
  - ▣ Extensibility is limited when considering upcoming
    - New assembly dialects...
    - New platforms...
    - New analysis techniques...
  
- Why?
  - ▣ Many tools are bolted onto an Intermediate Language (IL)
    - Information loss
    - Not sufficient to capture dialects (eg., HLASM)
  - ▣ Challenging to extend them to new analysis techniques
    - keep adding to IL?

# State of the Art Tools...

3

## IDA Pro



The screenshot displays the IDA Pro interface for a file named 'calc.idb (calc.exe)'. The main window shows assembly code for a function starting at address 01012475. The code includes instructions for setting up a stack frame, pushing arguments, and calling the `_XcptFilter` function. A comment indicates an exception filter for function 1012475.

```
assume fs:nothing, gs:nothing

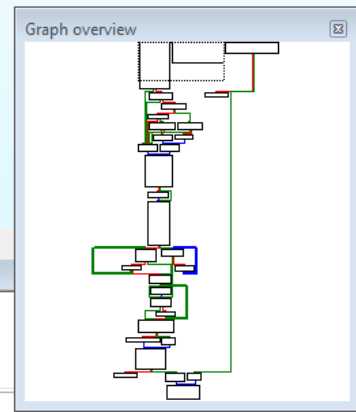
; Attributes: bp-based frame

public start
start proc near

Code= dword ptr -80h
var_7C= dword ptr -7Ch
StartupInfo= _STARTUPINFOA ptr -78h
var_34= dword ptr -34h
var_30= dword ptr -30h
var_2C= byte ptr -2Ch
var_28= byte ptr -28h
var_24= byte ptr -24h
var_20= dword ptr -20h
var_1C= dword ptr -1Ch
ms_exc= CPPEH_RECORD ptr -18h

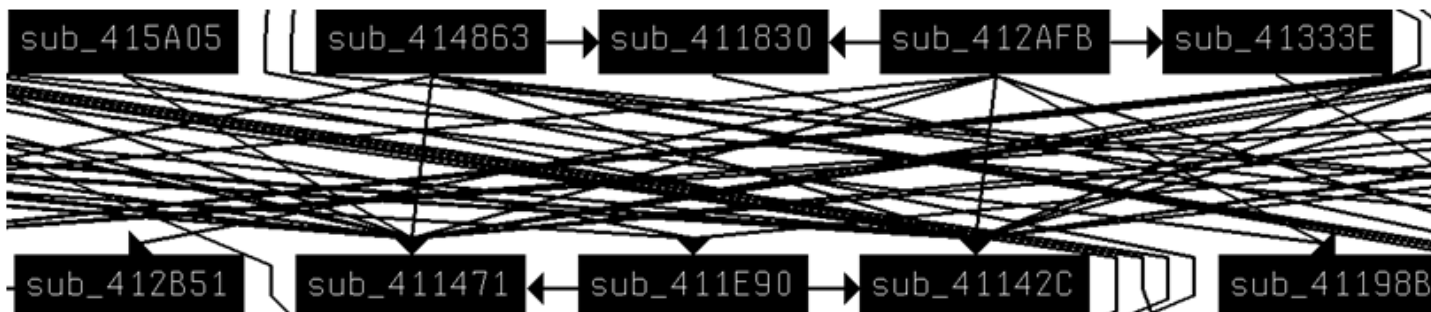
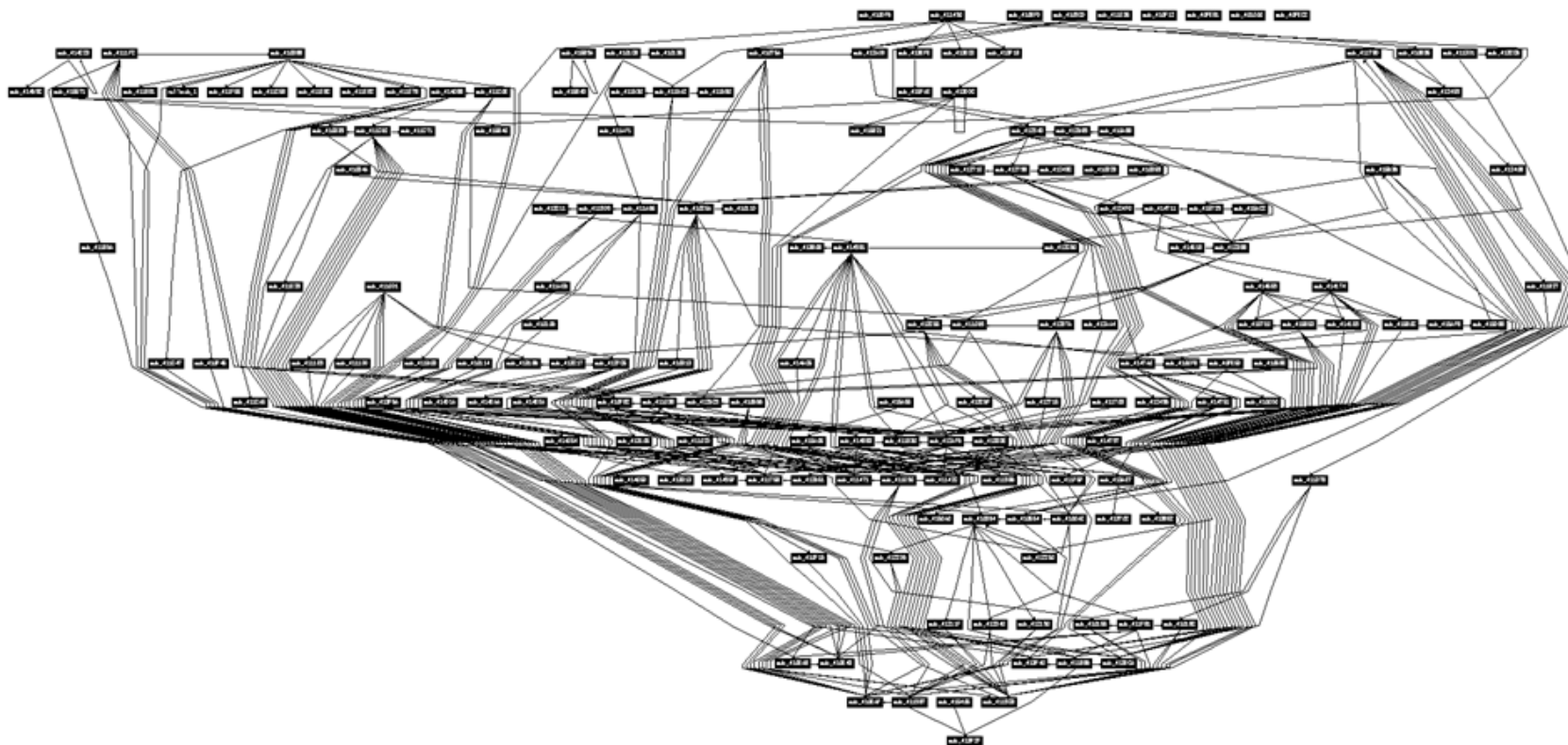
push    70h
push    offset stru_10015E0
call    _SEH_prolog
xor     eax, eax
ret     0
```

The left sidebar shows a list of functions, with `_XcptFilter` and `start` highlighted. The bottom status bar shows the current line (162 of 180) and various window settings. The bottom-most status bar indicates the system is idle and has 163GB of disk space.



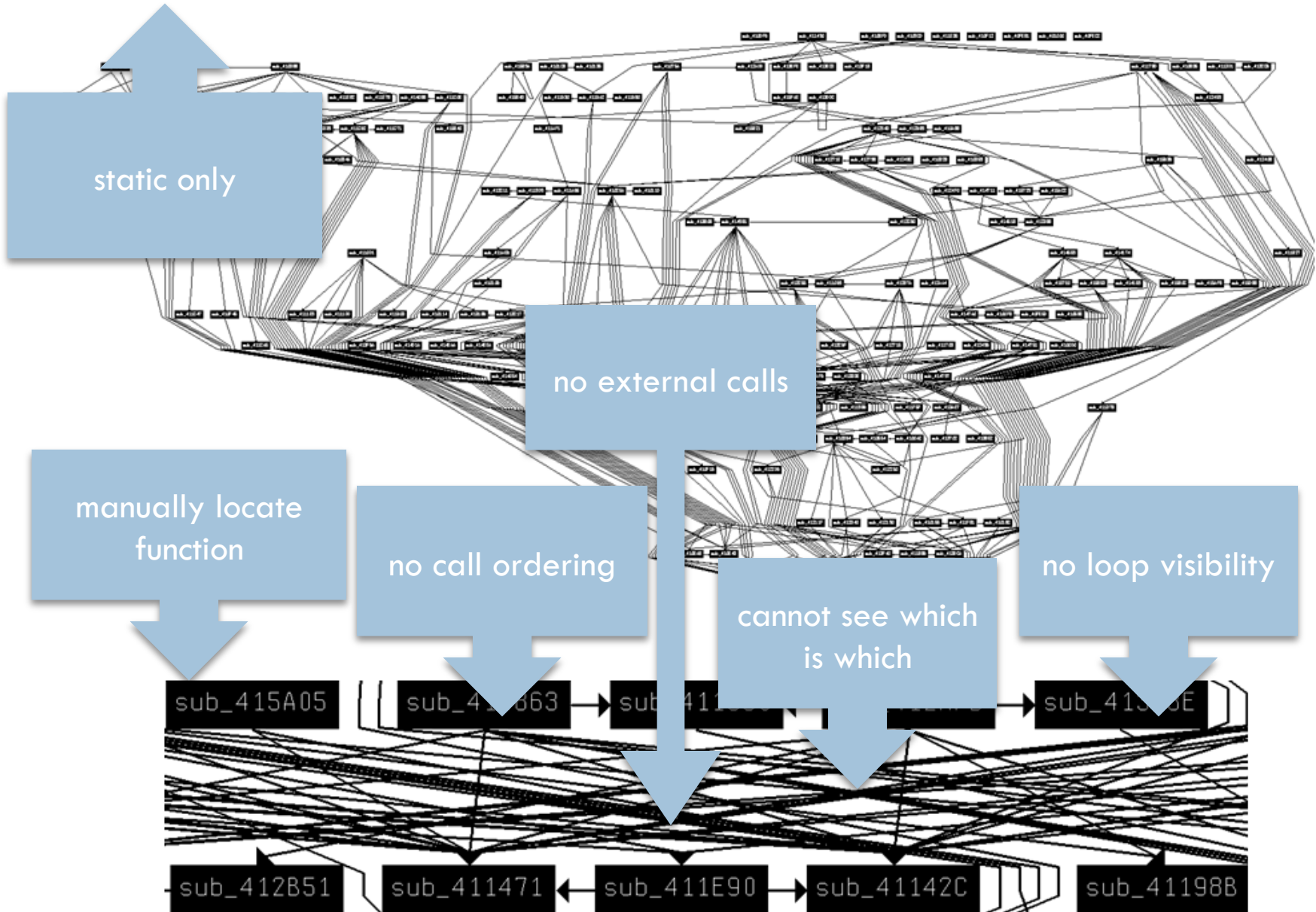
# Static Call Graph from IDA Pro

4

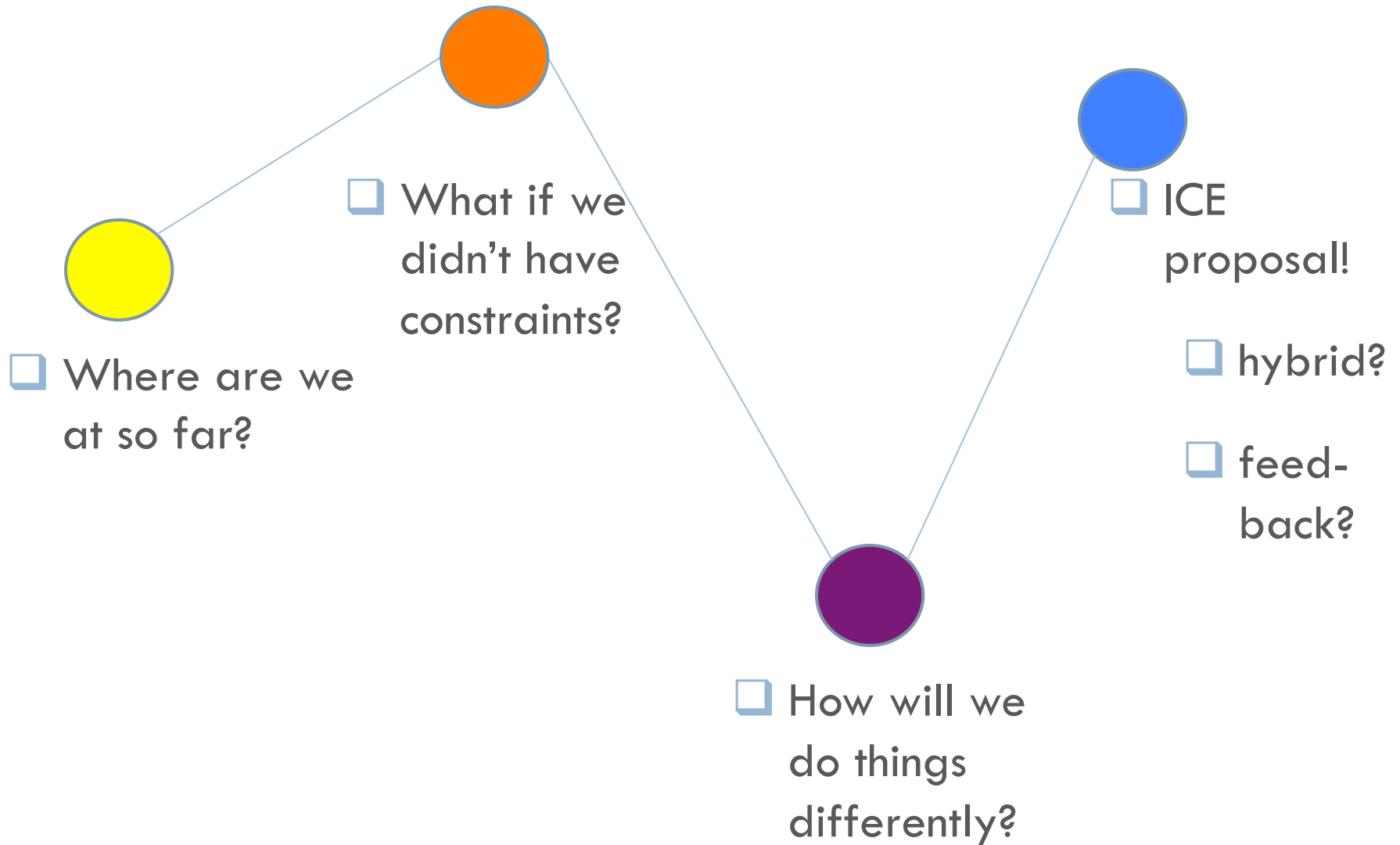


# Static Call Graph from IDA Pro

5



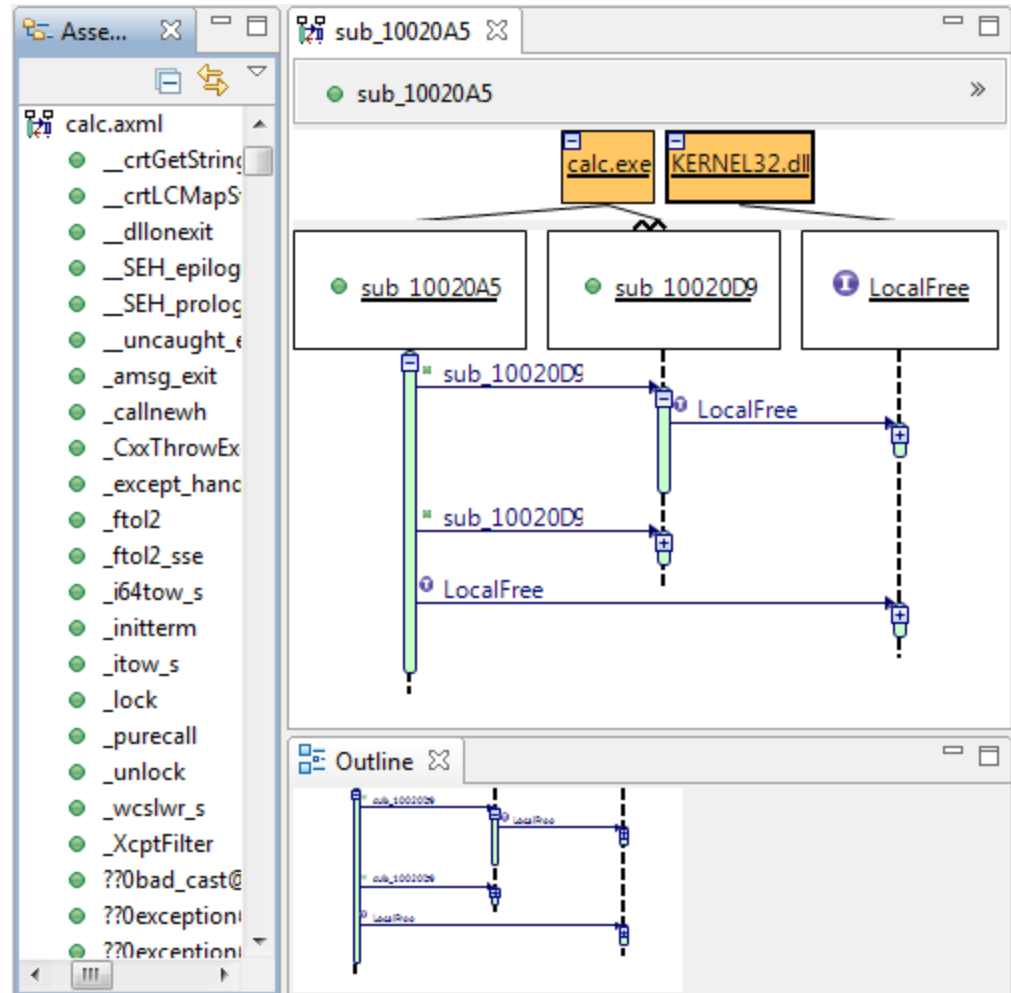
# Outline



Control Flow identified to be a major issue

Tracks

- Sequence diagram tool for control flow
- Eclipse plugin
- Integrated with IDA Pro
- 3 views:
  - Static
  - Dynamic
  - History



# Dynamic Control Flow View

8

- Actual control flow during program execution
  - ▣ Synchronized with IDA Pro during debugging sessions
  - ▣ Sequence diagram is built on the fly
- Trace file saved in XML, e.g.

```
<dynamicTrace rootexpanded="true">
  <root callindex="0" externalfile="User" module="User" name="User"/>
  <selection callindex="0" callingnode="loc_1429D97" externalfile="winlogon.exe"
    module="winlogon.exe"
    name="winlogon.exe"/>
  <call calladdress="FFFFFF" expanded="true" externalfile="winlogon.exe"
    functionaddress="FFFFFF"
    index="-1"
    module="winlogon.exe"
    name="tempname"
    root="false"/>
< /dynamicTrace>
```

# Tracks: Design and Implementation

9

← updateJavaServerPort 40000

← hello C:\mariposa.ose

← debug> 8: 10021FF sub\_10021FF calc.exe →

← rename sub\_10021FF sendMessage calc.exe →

← navigate> 8: 10021FF sub\_10021FF calc.exe →

← updateCursor 16852085

← setBreakpoint 16852085

← innerloop sub\_10021FF calc.exe →

← enableTracing / disableTracing →

← enableInner / disableInner

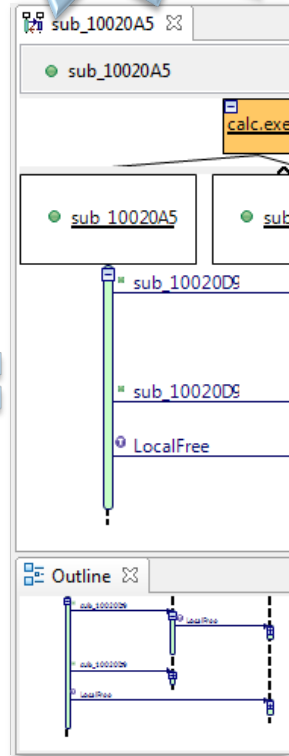
← prefCount 10

← stopLoop + addresses

← bye

← bye →

IDA Pro



# Tracks: Design and Implementation

← updateJavaServerPort 40000

hello C:\mariposa.ose

debug> 8: 10021FF sub\_10021FF calc.exe

rename sub\_10021FF sendMessage calc.exe

navigate> 8: 10021FF sub\_10021FF calc.exe

← updateCursor 16852085

← setBreakpoint 16852085

← innerloop sub\_10021FF calc.exe

← enableTracing / disableTracing

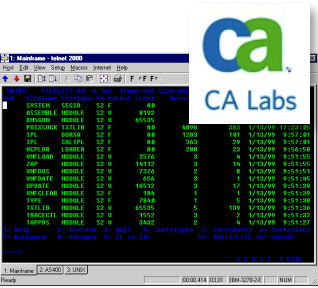
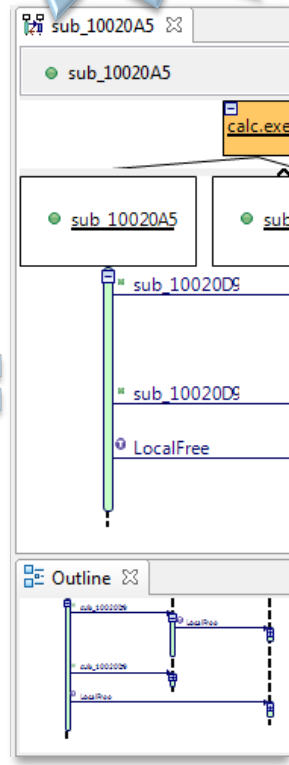
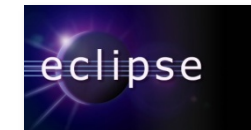
← enableInner / disableInner

← prefCount 10

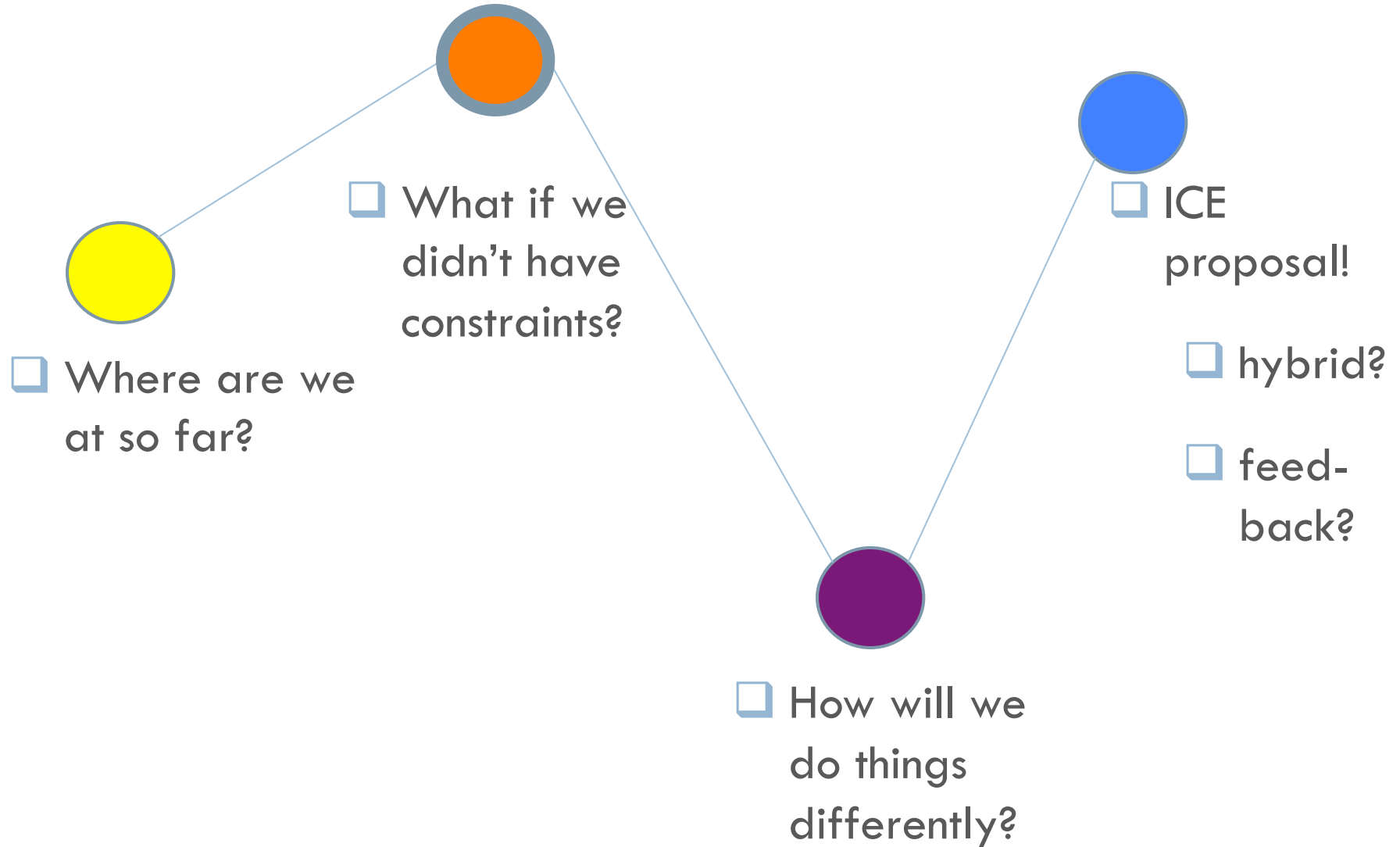
← stopLoop + addresses

← bye

← bye



# Outline



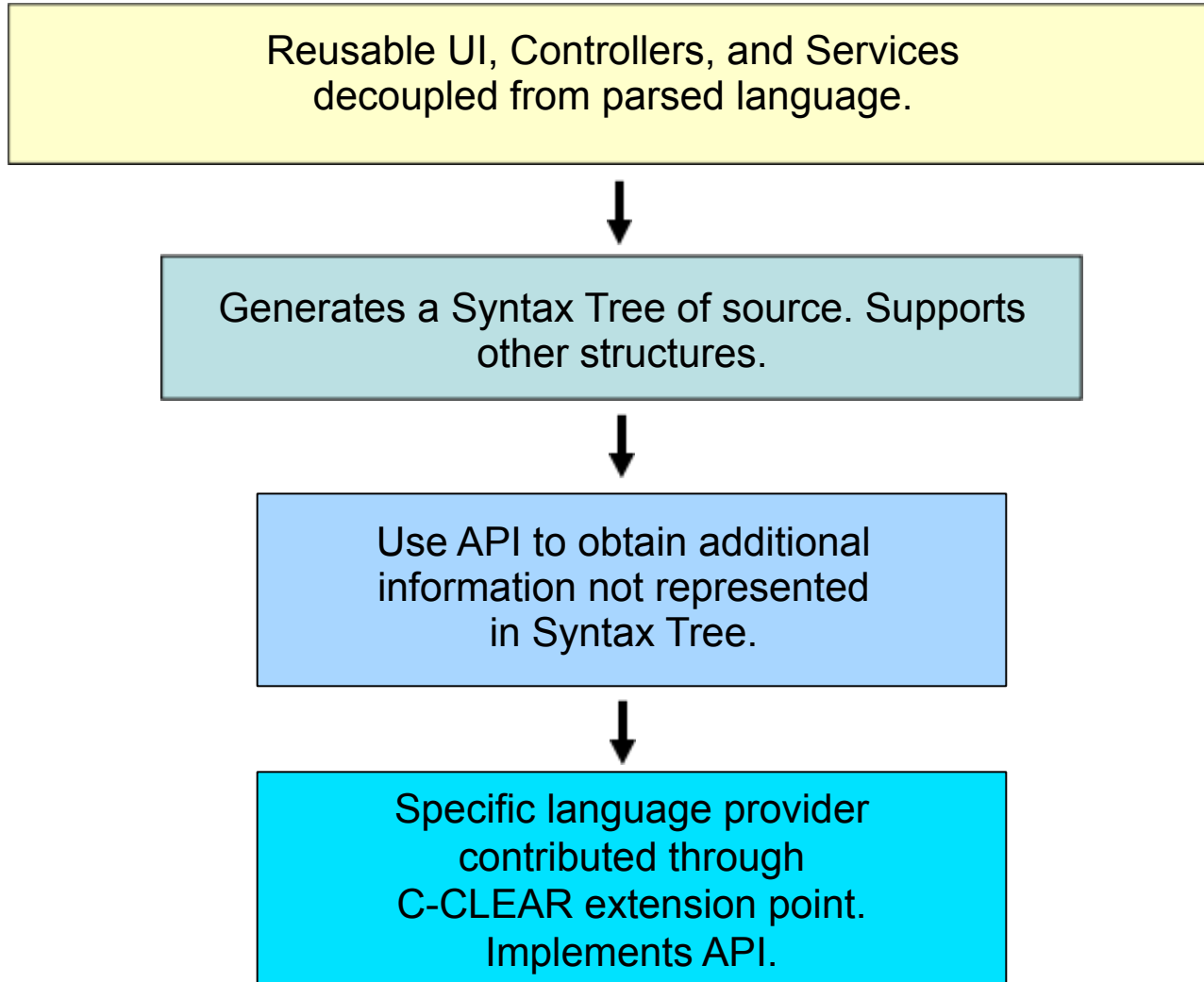
# Related Framework



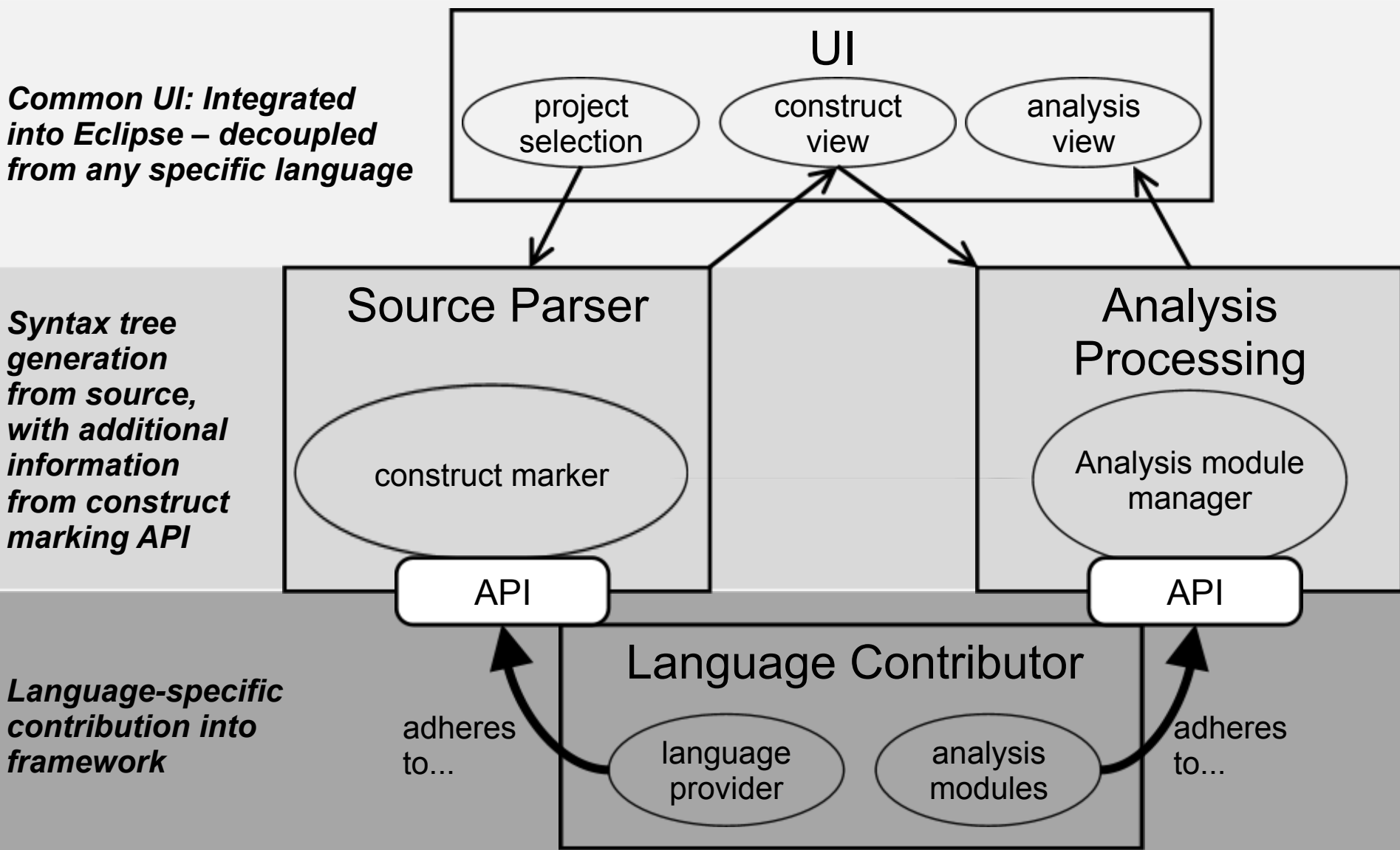
- A possible look into a parsing and analysis tooling framework that does not rely on a specific Intermediate Language: C-CLEAR
  - Extensible Eclipse-based tooling framework for high level language analysis
  
- Although meant for high-level analysis, some of the design components may be applicable to Assembly analysis frameworks
  - API to obtain language specific information not contained in an Intermediate Structure

# C-CLEAR API usage

13



# C-CLEAR design



# CPP Case Study

- ❑ Discover configuration CPP flags
- ❑ Discover syntactically similar code patterns - selected configurations
  - ❑ Use specific algorithms – select contributed queries
    - ❑ *Levenshtein* query
- ❑ IDE integration with Eclipse CDT/C Editor
  - ❑ Navigation to location

# C-CLEAR UI

The screenshot displays the Eclipse IDE interface with the C-CLEAR tool. The top editor shows the source code for `rcs.c`, with several `#ifdef` blocks for signals: `SIGHUP`, `SIGINT`, `SIGQUIT`, `SIGPIPE`, and `SIGTERM`. The `SIGHUP` block is highlighted with a red box. A red arrow points from this box to the `C-CLEAR Query Results` table.

The `C-CLEAR Constructs` panel on the left shows a list of constructs with checkboxes. The constructs `SIGABRT`, `SIGHUP`, `SIGINT`, `SIGPIPE`, `SIGQUIT`, and `SIGTERM` are checked and highlighted with a red box. A red arrow points from this box to the `C-CLEAR Query Results` table.

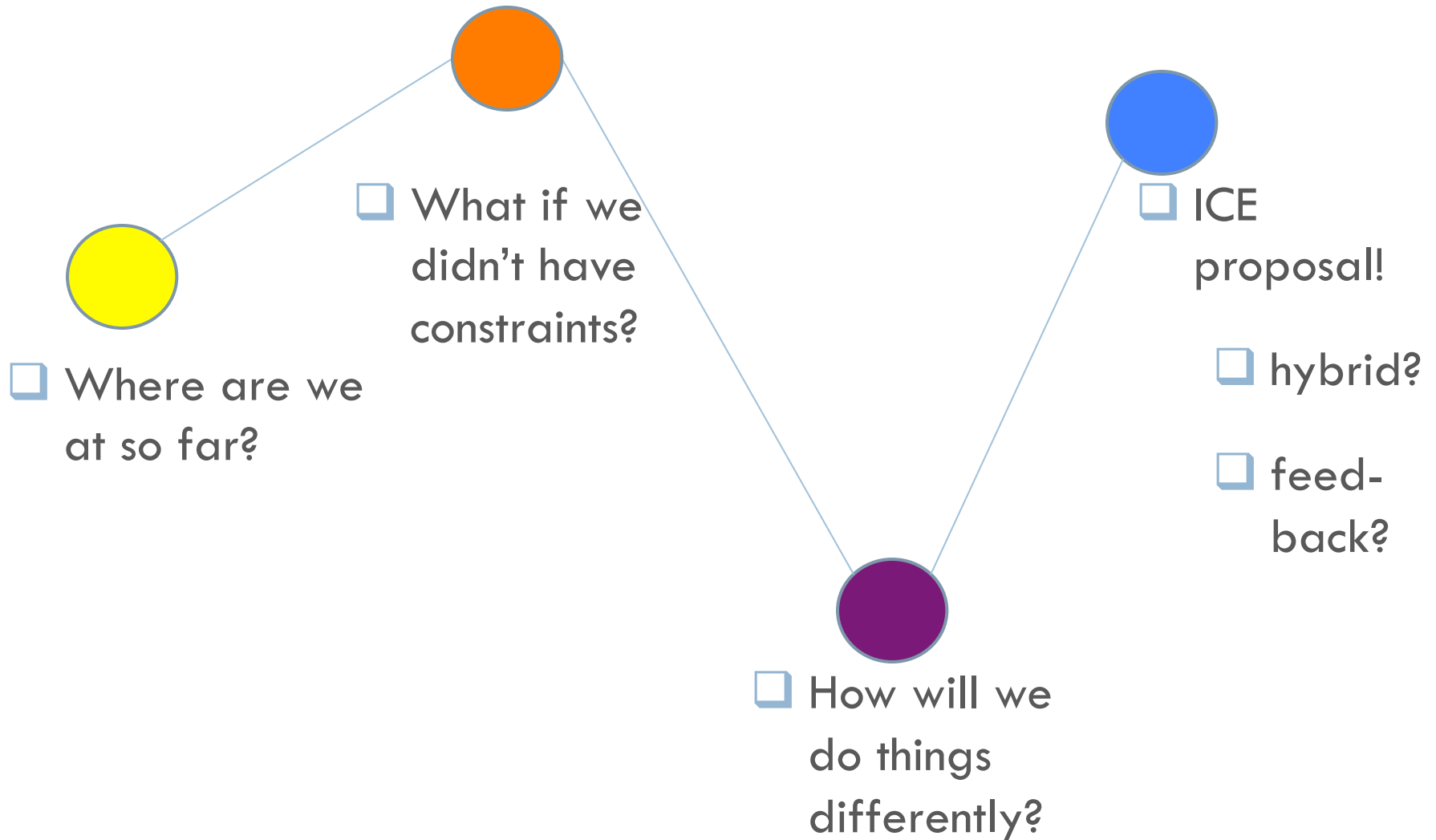
The `C-CLEAR Query Results` table shows the following data:

RESULTS	CONSTRUCTS	LOCATIONS	FILES
<code>(void)SIG_register(SIGABRT,patch_cleanup);</code>	6	18	3
<code>CVS source/src/patch.c</code>	SIGINT, SIGABRT, SIGPIPE, SIGHUP, SIGQUIT, SIGTERM	6	
<code>CVS source/src/main.c</code>	SIGINT, SIGABRT, SIGPIPE, SIGHUP, SIGQUIT, SIGTERM	6	
<code>CVS source/src/rcs.c</code>	SIGINT, SIGABRT, SIGPIPE, SIGHUP, SIGQUIT, SIGTERM	6	
8356	SIGABRT		
8359	SIGHUP		
8362	SIGINT		
8365	SIGQUIT		
8368	SIGPIPE		
8371	SIGTERM		
<code>ptrend=rcsbuf-&gt;ptrend;</code>	0	10	1
<code>(ptr=rcsbuf_fill(rcsbuf,ptrend,keyp,vaip))if(ptr==NULL)</code>	0	5	1
<code>caseSIGABRT:name="abort";break;</code>	4	4	1
<code>*/</code>	0	3	2
<code>(ptr=rcsbuf_fill(rcsbuf,ptr,keyp,(char*)NULL))if(ptr== [.]</code>	0	3	1
<code>(ptr=rcsbuf_fill(rcsbuf,ptr,(char*)NULL,(char*)NULL); [.]</code>	0	2	1

# Case Studies

- Building on Tracks
  - ▣ XML: contains metadata for both dynamic and static analysis
    - Integration with other tools
    - Re-use C-CLEAR UI and other controller components
      - Integration with tools where UI extensibility is not possible
- Variants of String Copy
  - ▣ Build a framework where different analysis modules are contributed
    - Levenshtein

# Outline



## Evaluation Criteria:

- ❑ Extensibility
- ❑ Platform Independence
- ❑ Static Analysis Support
- ❑ Dynamic Analysis Support
- ❑ Instruction Set Architectures (ISA's) Supported
- ❑ Instruction Set Completeness
- ❑ Ease of Use and Documentation



## BitBlaze

- Modular design
  - Clear separation between modules:
    - IL generation and Static analysis (VINE)
    - Dynamic Analysis (TEMU)
    - Combined Analysis Module (Rudder)
- Biendian support mechanism
- Built-in IL-to-C translator



## IDA Pro

- Extensible plug-in architecture
- Many supported ISA's
- Simplified version of disassembled binary to improve comprehensibility



## BinNavi

- Clean, attractive GUI
- Built-in static visual analysis
- The design of the IL (REIL) employs multiple program comprehension techniques and innovations
  - Infinite machine model
  - Every instruction has exactly 3 operands
  - Instructions are all one byte in size
  - REIL instructions are easy to map back to the address of the original instruction
    - Ex: 0x00CF12.00



## New Additions

- Mechanism to identify common blocks of code (Ex: strcpy)
  - Colour-coded highlighting
  - Code marking and filtering
  - Code folding



## BitBlaze

- Add support for multiple architectures
- Reduce or eliminate dependence on third party components, especially during IL generation
- Add a GUI and visual analysis toolset



## IDA Pro

- Increase tool integration and interoperability
- Reduce or eliminate dependency on third-party components for dynamic analysis
- Use microcode as an IL for tool development as well to increase comprehensibility of the disassembled code
- Add visual analysis tools for dynamic analysis
  - Does IDA currently have any?



## BinNavi

- Add a mechanism to provide biendian support
- Rework IL into a form that can support dynamic analysis

# A Fork in the Road...

27

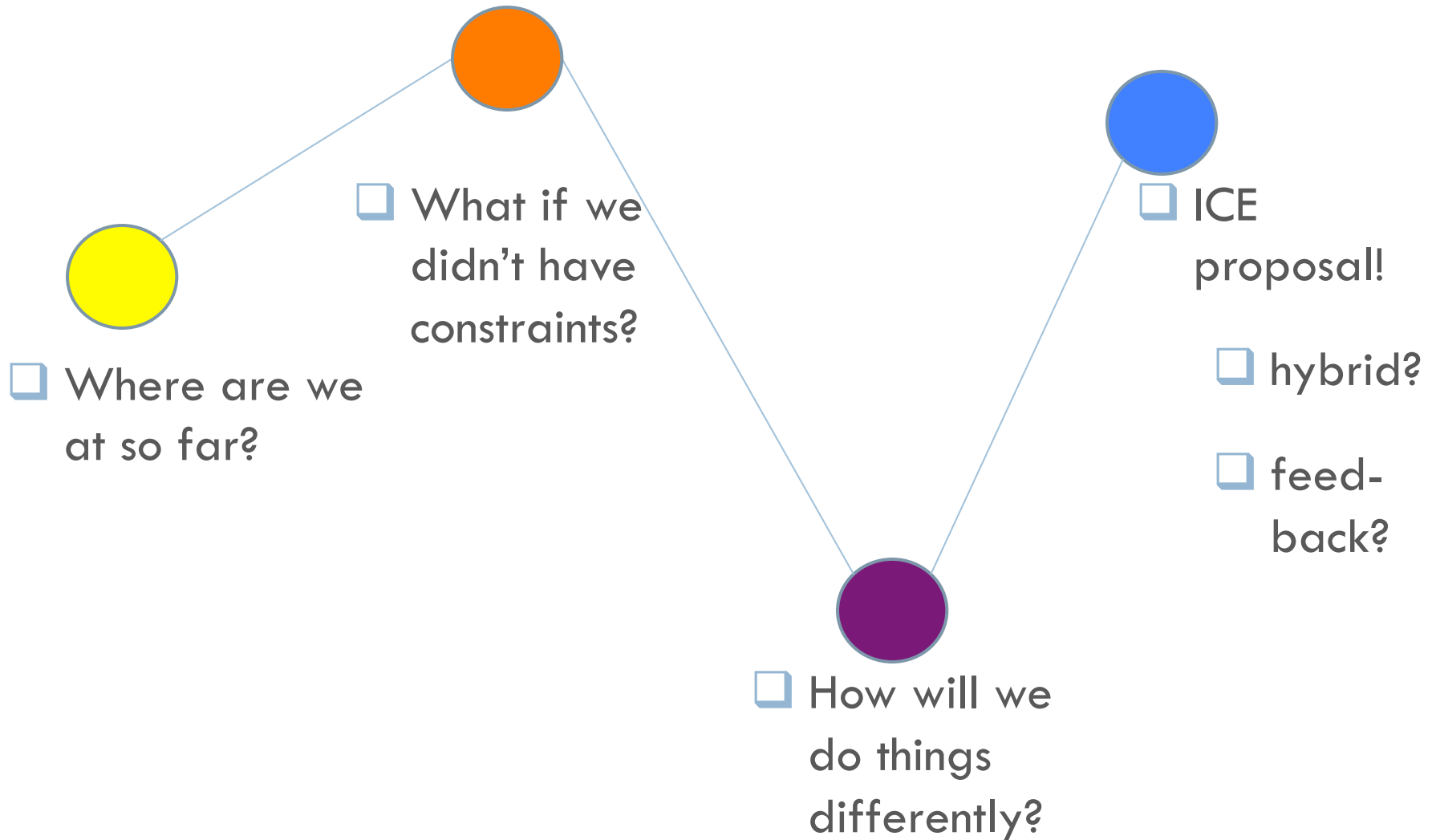
## Platform Independence

- ❑ Binaries from supported ISA's can be analysed regardless of host and source platform
- ❑ No support for kernel-level instructions

## Multiple Platform Support

- ❑ Can directly execute IL to perform dynamic analysis
- ❑ Can develop tools for platform-specific vulnerabilities
- ❑ Must build in extensibility mechanism

# Outline





- Enable sophisticated analyses
  - Static
  - Dynamic
- Extensible
  - Instruction set architecture
  - Analyses
  - UI
- OS Independent

# Ways To Get There



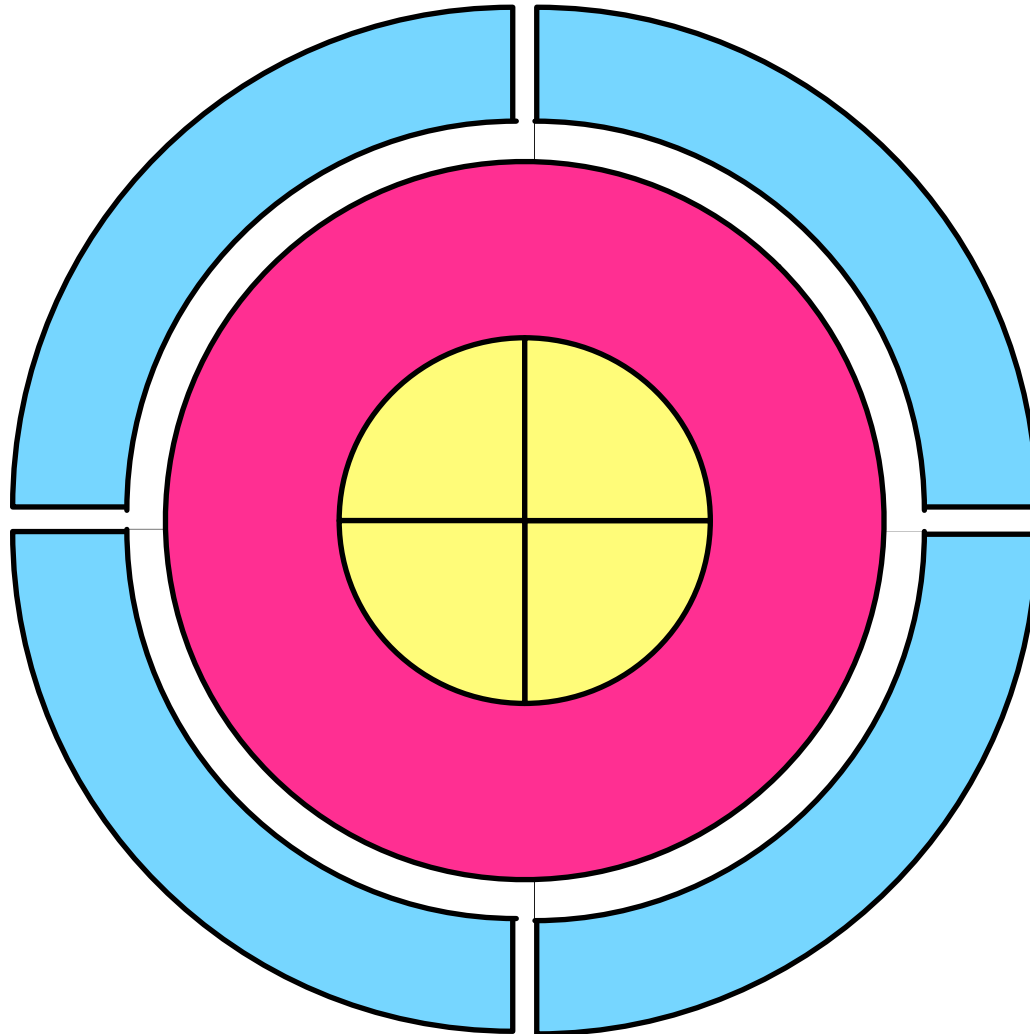
30

- Intermediate Language
- Framework
- Hybrid

# Intermediate Language



31



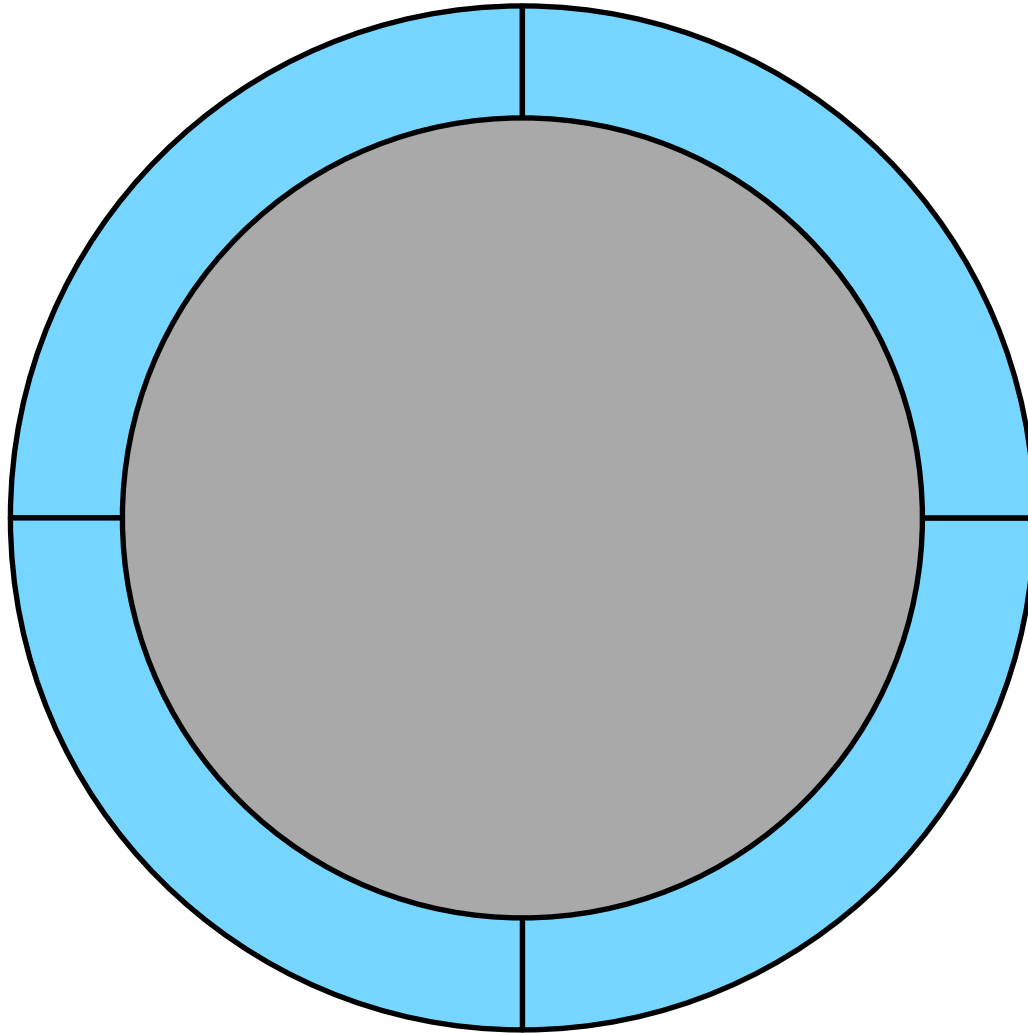
Analysis



Intermediate  
Language

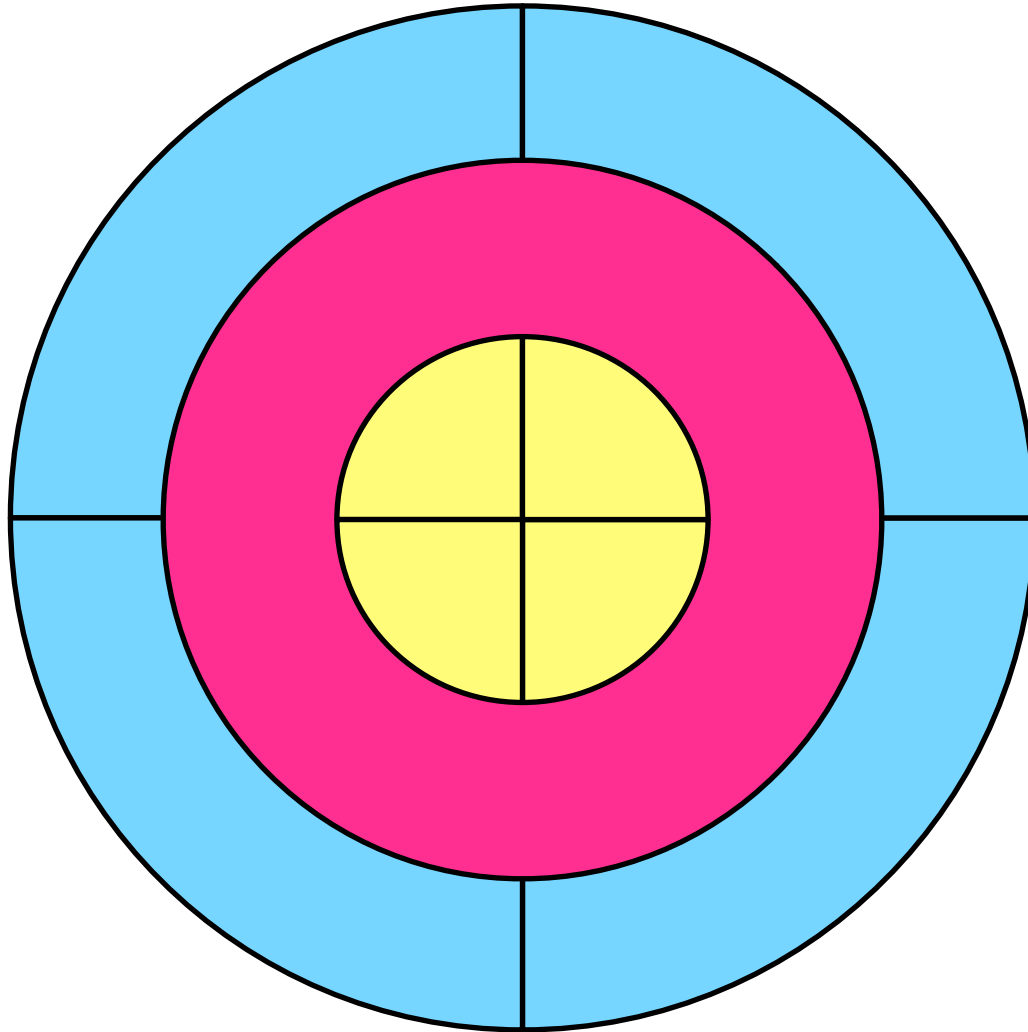


ISA



 Analysis

 Binary



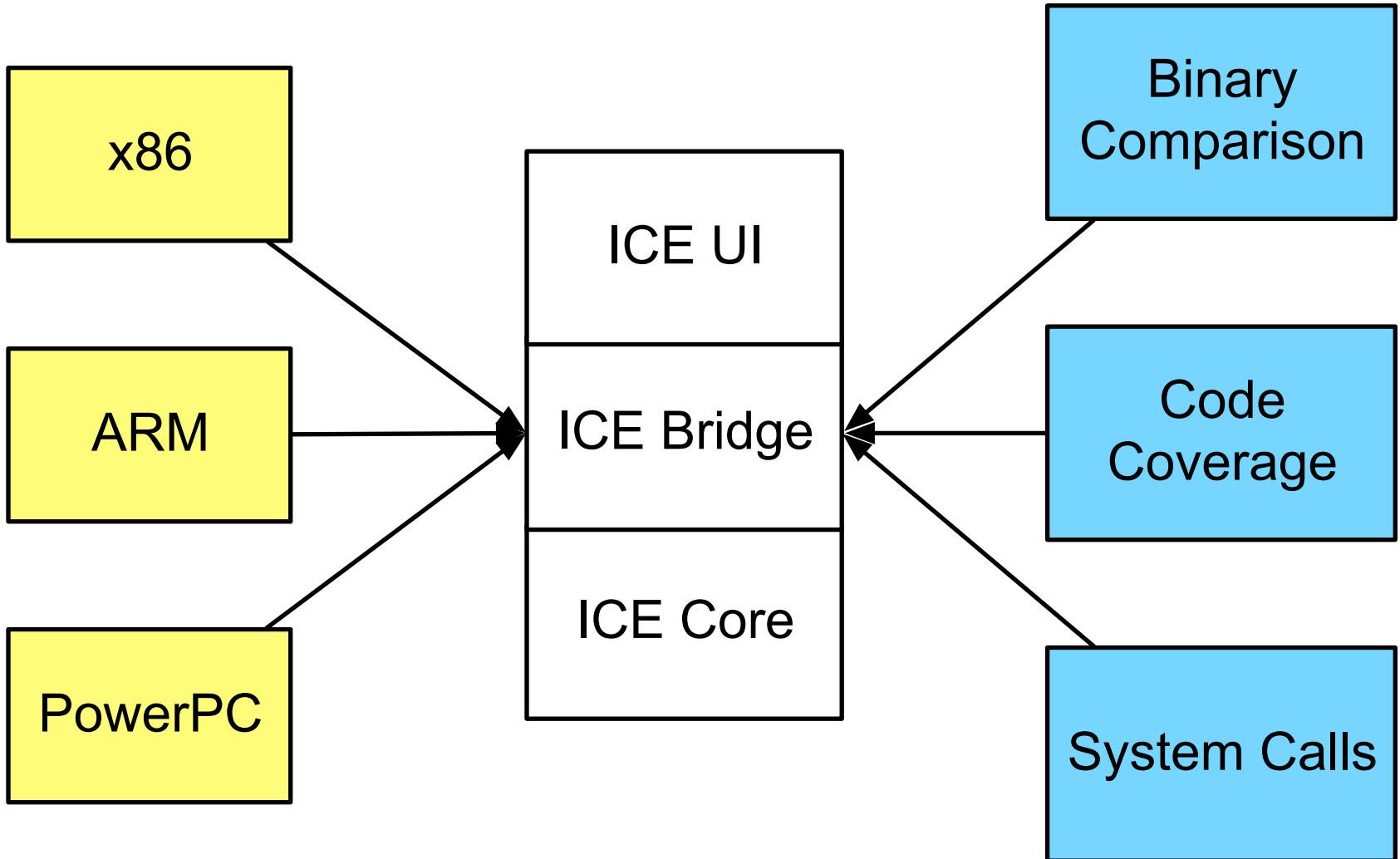
Analysis

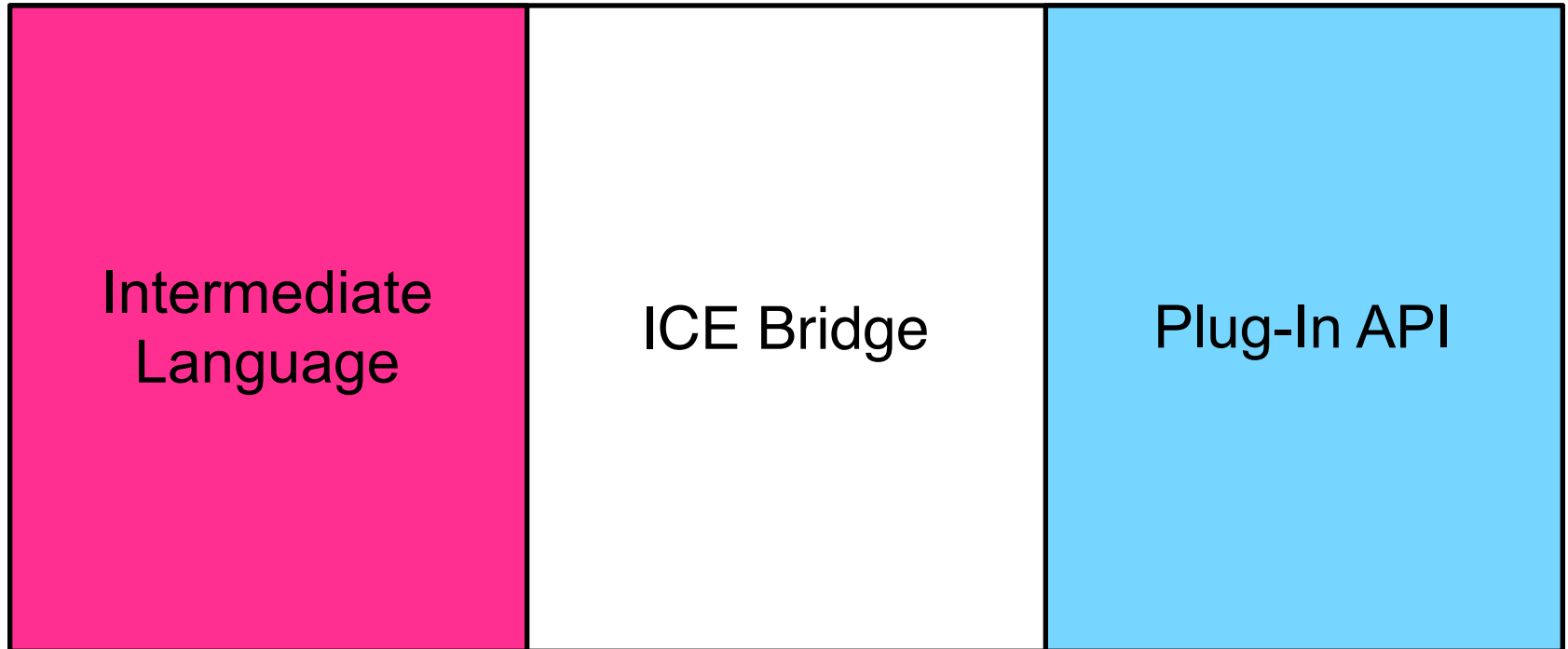


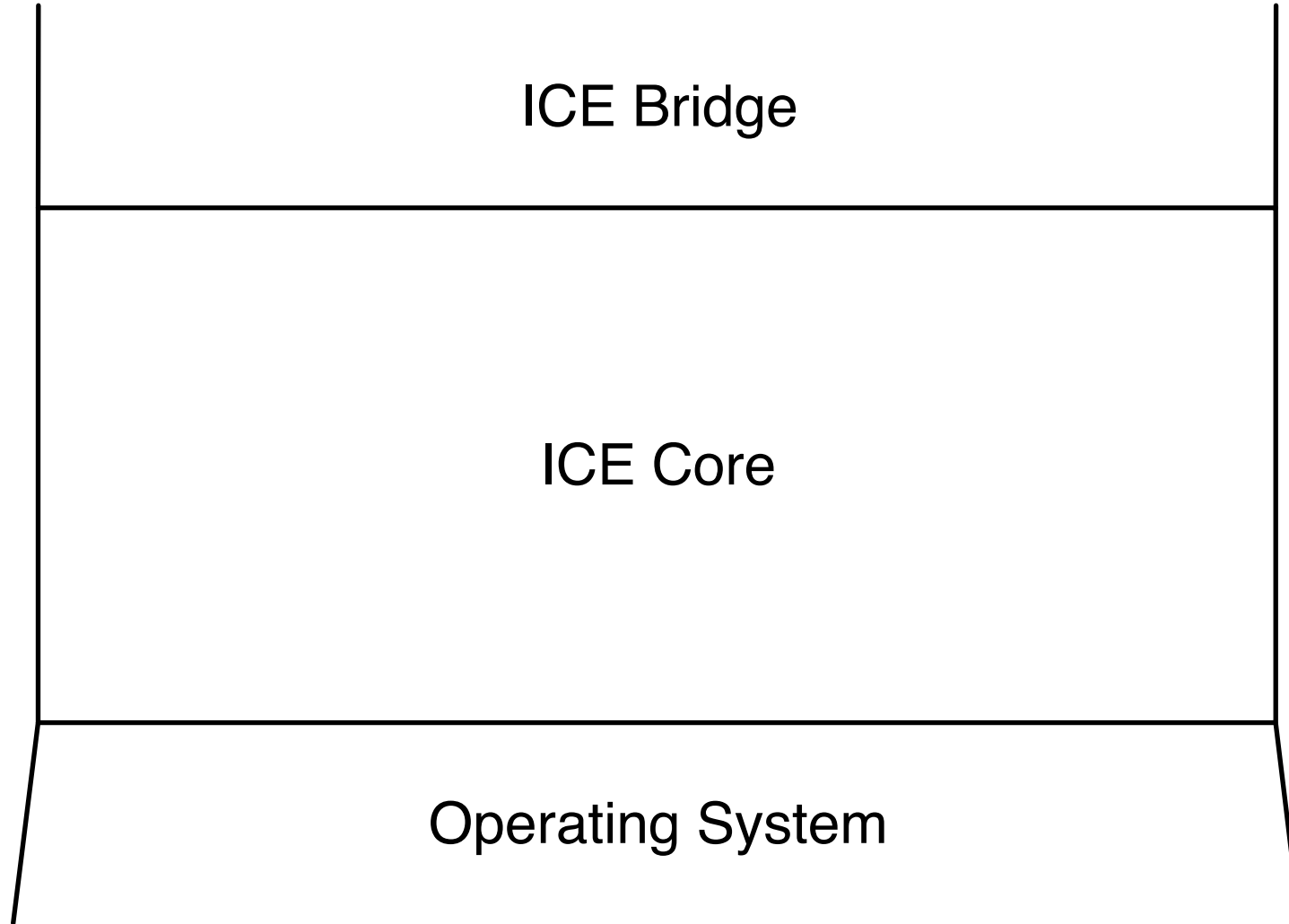
Intermediate  
Language



ISA







# Where To Next?



37

- Deeper exploration of IDA Pro and BinNavi
- Determine set of base analyses
- Investigate options for dynamic analysis
- Proof-of-concept based on REIL and our own API

38

Questions?

Thank you!