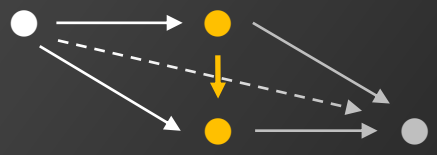
Part 4 Draft

Windows API

for Software Diagnostics

Accelerated

With Category Theory in View



Dmitry Vostokov
Software Diagnostics Services

API and Errors

- Windows protocols
- Windows error codes reference
- Thread Information Block
- Win32 values
- NTSTATUS values
- HRESULT values

Exercise W8

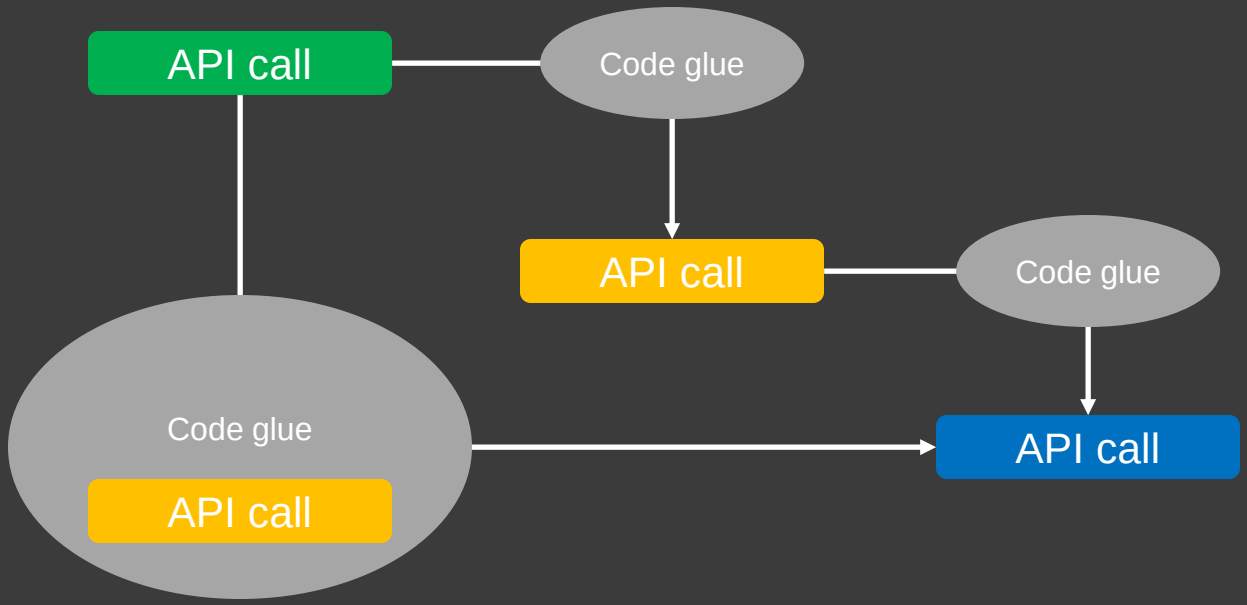
- **Goal:** Explore different Windows API error types
- **Memory Analysis Patterns:** Last Error Collection
- **ADDR Patterns:** Function Skeleton; Call Path; Structure Field
- \AWAPI-Dumps\Exercise-W8.pdf

Windows API Formalization

Ideas from Conceptual Mathematics

API Compositionality

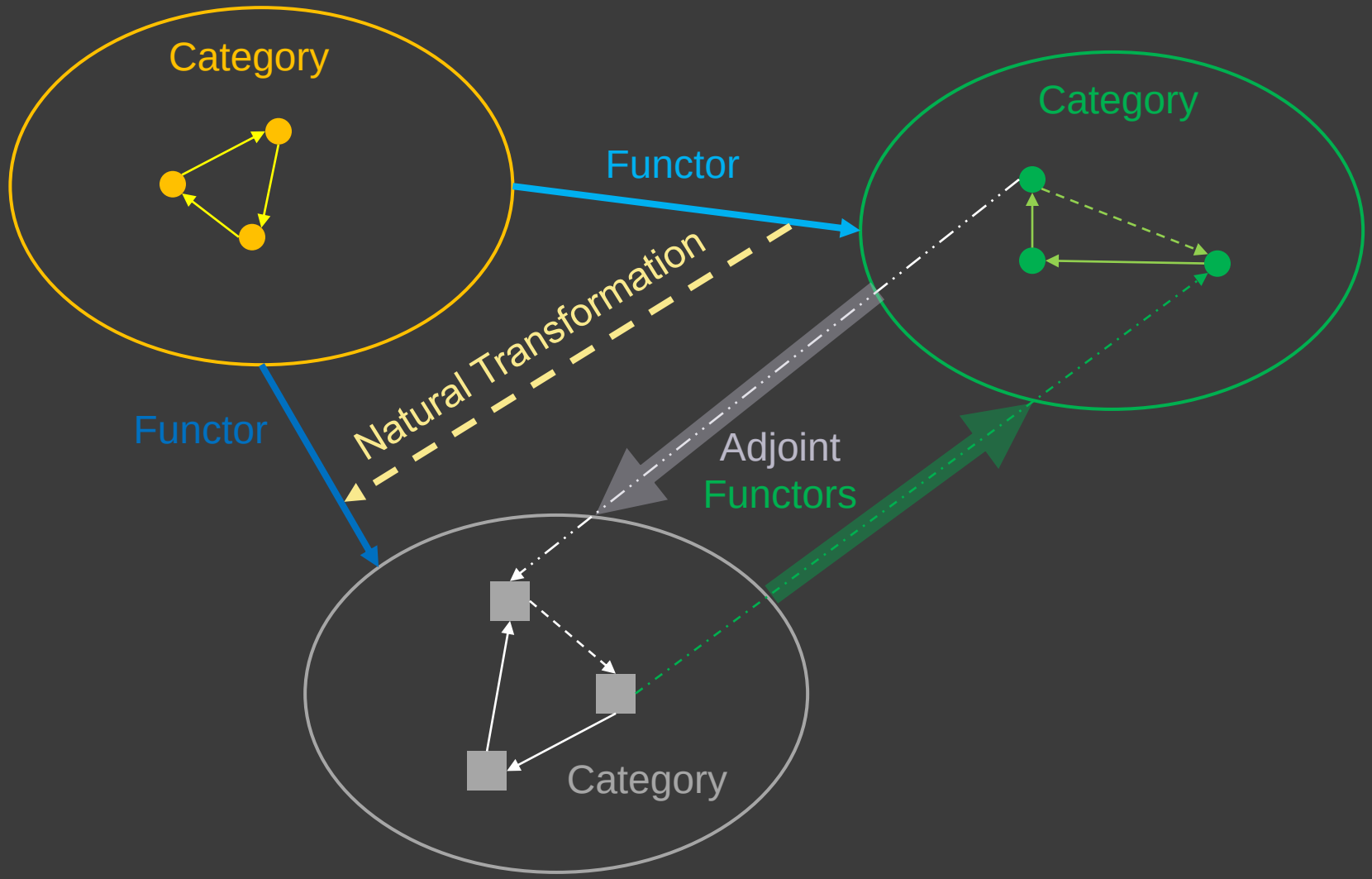
- Principle of compositionality



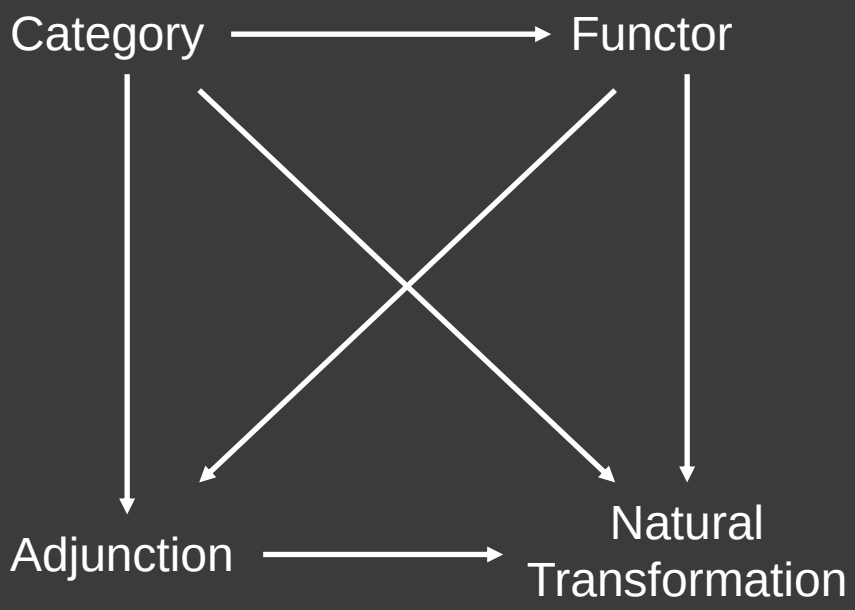
Category Theory Language

- Category
  - Objects
  - Arrows between objects (must be transitive, if A → B and B → C then A → C)
- Functor
  - Arrow between categories (can be the same category)
  - Maps objects to objects and arrows
- Natural Transformation
  - Arrows between functors in a category of functors
- Adjunction
  - Relationship between functors, change of perspective, back translation

A View of Category Theory

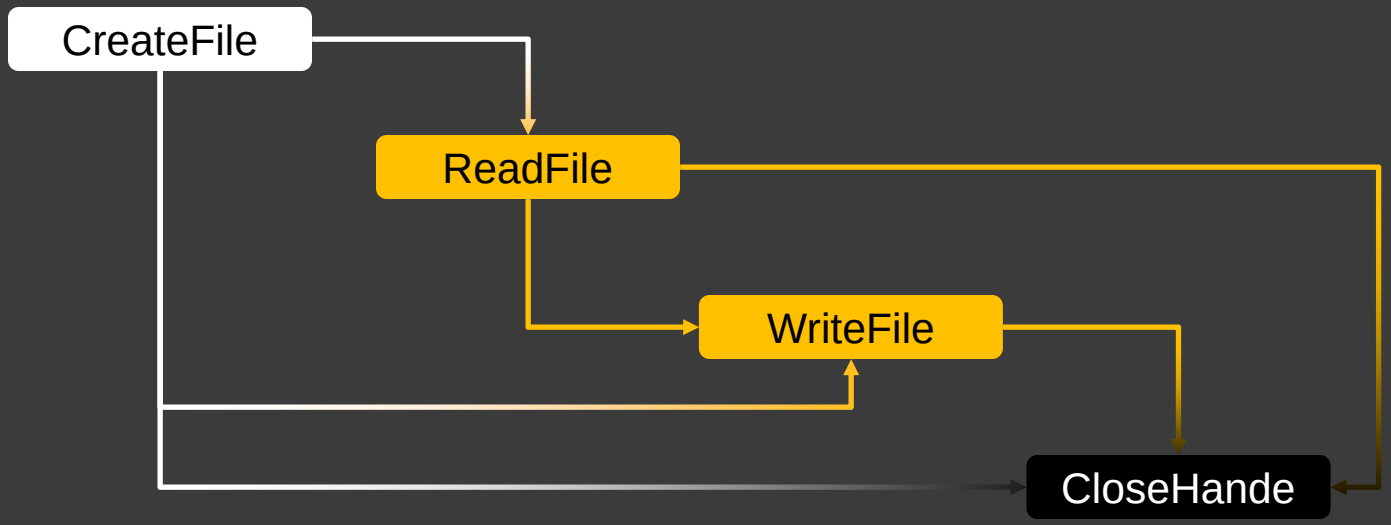


Category Theory Square



API Category

- API as objects, glue code as arrows
- API as arrows, glue code as objects fails at composition
- Initial and terminal API objects in subcategories

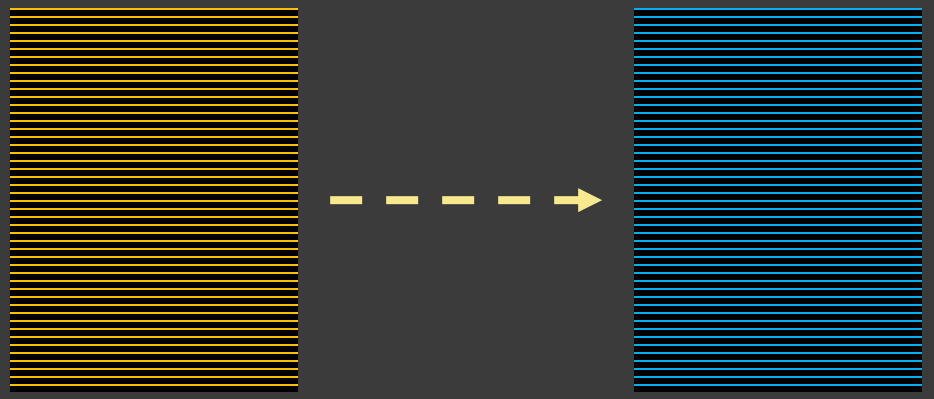


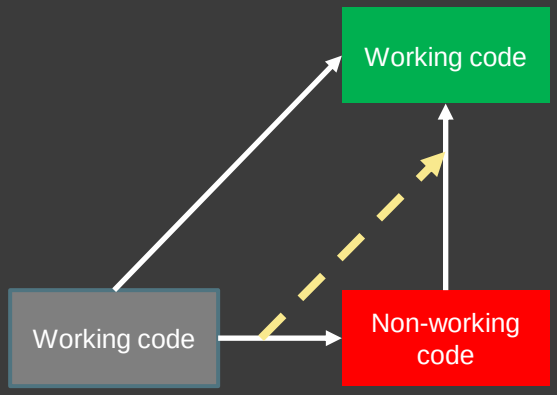
API Functor

- Translates between API layers (different API)
- Stack trace as functor
- Translates between different API sequences
- Endofunctor – between the same API
- Translates between different code implementations

API Natural Transformation

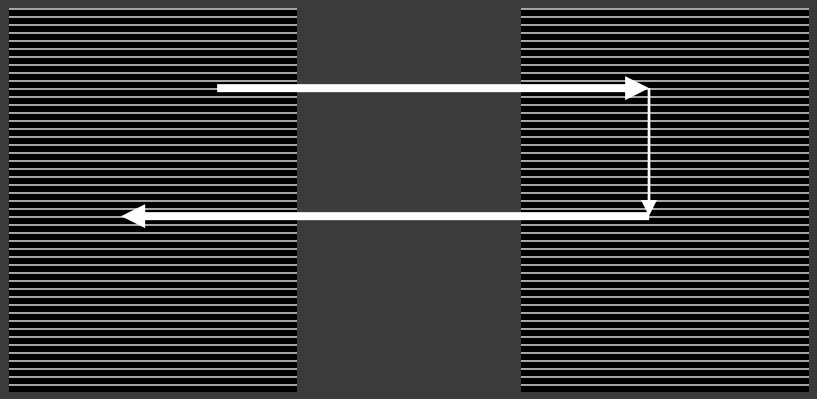
- Maps between different vertical API sequences (stack traces)
- Maps between different code translations
- Diagnostics and debugging as natural transformation

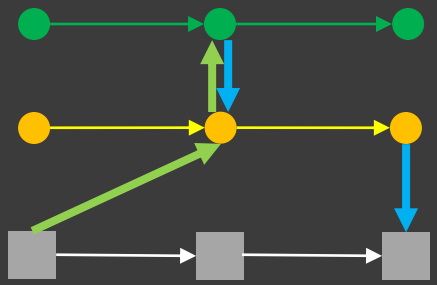




API Adjunction

- Navigation between different API sequences
- Call and return stack trace sequences, callbacks (when stack traces correspond to vertical API sequences)
- Back translation between traces/logs (when traces correspond to API horizontal sequences)





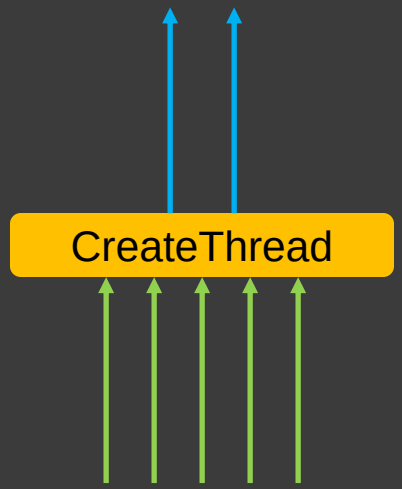
Informal n-API

- Arrows between arrows
- 1-API – normal API usage
- 2-API – diagnostics, debugging
- 3-API – higher diagnostics, debugging (debugging the debugging)
- ∞-API – for homework ☺

API I/O

- Categories – one input, one output
- Operads – many inputs, one output
- Properads – many inputs, many outputs

```
HANDLE CreateThread(
  [in, optional]  LPSECURITY_ATTRIBUTES   lpThreadAttributes,
  [in]            SIZE_T                  dwStackSize,
  [in]            LPTHREAD_START_ROUTINE  lpStartAddress,
  [in, optional]  __drv_aliasesMem LPVOID lpParameter,
  [in]            DWORD                   dwCreationFlags,
  [out, optional] LPDWORD                 lpThreadId
);
```



Windows API and Languages

API and C#

- P/Invoke example:

```
using System;
using System.Runtime.InteropServices;

class WapiTest
{
  [DllImport("user32.dll")]
  public static extern uint MessageBox(ulong hWnd, string message, string title, uint flags);

  public static void Main()
  {
    MessageBox(0, "Hello Windows API!", "From C#", 0);
  }
}
```

API Metadata

- Overview
- C#/Win32 P/Invoke Source Generator

API and Scala Native

- Documentation
- Scala Native
- Native code interoperability

Exercise W9

- **Goal:** Install Scala Native environment and write a simple program that uses Windows API

- \AWAPI-Dumps\Exercise-W9.pdf

API and Golang

- windows package
- Example:

```
package main

import "unsafe"
import "golang.org/x/sys/windows"

func main() {
  var user32 = windows.NewLazyDLL("user32.dll")
  var procMessageBox = user32.NewProc("MessageBoxW")

  message, _ := windows.UTF16PtrFromString("Hello Windows API!")
  title, _ := windows.UTF16PtrFromString("From Golang")
  procMessageBox.Call(0, uintptr(unsafe.Pointer(message)), uintptr(unsafe.Pointer(title)), 0)
}
```

API and Rust

- Rust for Windows
- Example:

```
use windows_sys::{
  core::*, Win32::UI::WindowsAndMessaging::*
};

fn main() {
  unsafe {
    MessageBoxW(0, w!("Hello Windows API!"), w!("From Rust"), 0);
  }
}
```

API and Python

- ctypes library
- Example:

```
from ctypes import windll

windll.user32.MessageBoxW(0, "Hello Windows API!", "From Python", 0)
```