What if we could reduce the tedium of Reverse Engineering?

(aka "How I achieved my childhood dream of doing dinosaur-related research")

TRex: Practical Type Reconstruction for Binary Code

Jay Bosamiya[1], Maverick Woo[2], and Bryan Parno[2]

[1]Microsoft Research
[2]Carnegie Mellon University

Problem

> Legacy software built across years
  > Broken, insecure, often source-unavailable

> Decompilers help understand program behavior

> In practice, still require lots of manual input
  > Despite decades of engineering and academic research

> This work: focusing on types

Decades of Research, yet…

Source "Ground Truth" C

```
struct Node {
  int data;
  struct Node* next;
};
```

Machine Code

```
… DWORD …
… QWORD PTR …
```

Decades of Research, yet…

Source "Ground Truth" C

```
struct Node {
  int data;
  struct Node* next;
};
```

Compile

Machine Code

```
… DWORD …
… QWORD PTR …
```

Decompile

Ghidra 

```
undefined4 ?
undefined4 * X
```

Decades of Research, yet…

Source "Ground Truth" C

```
struct Node {
  int data;
  struct Node* next;
};
```

Compile

Machine Code

```
… DWORD …
… QWORD PTR …
```

Decompile

Ghidra 

```
undefined4 ?
undefined4 * X
```

Decompile

IDA Pro/HexRays 

```
__int64 X
unsigned int * X
```

On the Impossibility of Ground Truth

> Prior works, both Type Recovery (deductive) and Type Prediction (ML-based) attempt to recover *the* ground-truth
  > Tempting: types existed before, "just" recover them

> Implicit assumption: a perfect recovery tool ζ can recover source types γ (modulo renaming ρ)
  > $\forall S, v \in vars([S]).\ \rho\big(\zeta([S], v)\big) = \rho(\gamma(S, v))$

> We prove this assumption is fundamentally flawed; details in paper

Type Reconstruction: Capturing Behavior

> Perfect recovery is impossible

> What do reverse engineers really want?

> Claim: rather than perfect recovery, reverse engineers want to understand behavior

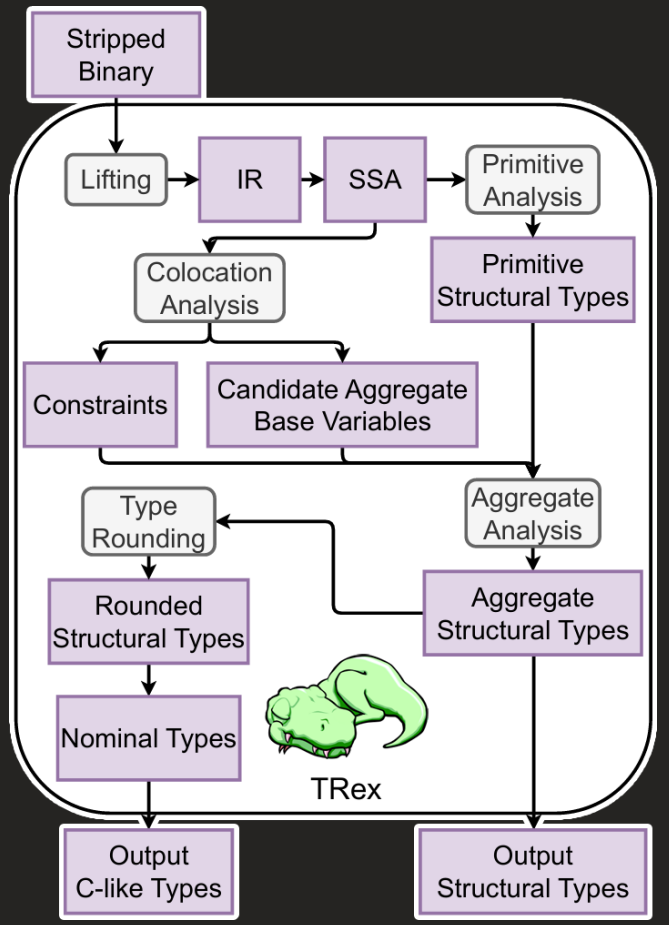
> **Key Insight:** capture observable behavior

Our Categorization of Automated Type Inference

Type…	Multi-language	No GT Assumption	Technique
Inference		N/A	Deductive
Prediction	✓		Learning
Recovery	✓		Deductive
Reconstruction	✓	✓	Deductive

TRex: Type Reconstruction

- Deductive constraint-based technique
- **Key insight:** use structural types until as late in analysis as possible
- Supports outputting all C scalar types, pointers, structs, unions, arrays, …
  - Including recursive types!



Structural Types Capture Behavior

- De facto language for decompiler output is C
  - Thus, tempting to use C-like nominal types
  - But that leads to need for complex mechanisms like in prior work

- Instead, Structural Types are most natural to task of Type Reconstruction
  - Capture behaviors precisely, even if not expressible in C
  - “Type rounding” to C-like types in final stages

Example TRex Phase: Type Rounding

```
StructuralType {
  COPY_SIZES  {8, 32}
  INTEGER_OPS {Add_32, Sub_32, SDiv_32}
  POINTER_TO  None
  …
}
```

(Equivalent to C int32_t)

```
StructuralType {
  COPY_SIZES  {8, 16, 32}
  INTEGER_OPS {Add_32, Sub_32,
               Mul_32, SDiv_32, …}
  POINTER_TO  None
  …
}
```

Lots More in the Paper!

We show:

> Ground Truth is Provably Impossible

> Type Rounding is NP-hard

> How to safely manage non-conservative reasoning

> Designing a suitable metric despite ground truth being impossible

> …more…

Qualitative Evaluation

On a Singly Linked List Example

```
struct Node {
  int data;
  struct Node* next;
};
```

- TRex: ✓ correct (Node*)
- Ghidra: ✗ incorrect (undefined4*)
  - Widely used by practitioners
  - Similar results for other widely used tools: Hex-Rays (unsigned int*) and Binary Ninja (int32_t*)
- ReSym: ? non-deterministic results
  - Distinguished paper award at CCS'24; ML-based
  - 2%: ✓ correct
  - 36%: ✗ equiv. to a valid-but-incorrect C type
  - 62%: ✗ no possible equiv. valid C type

Quantitative Evaluation

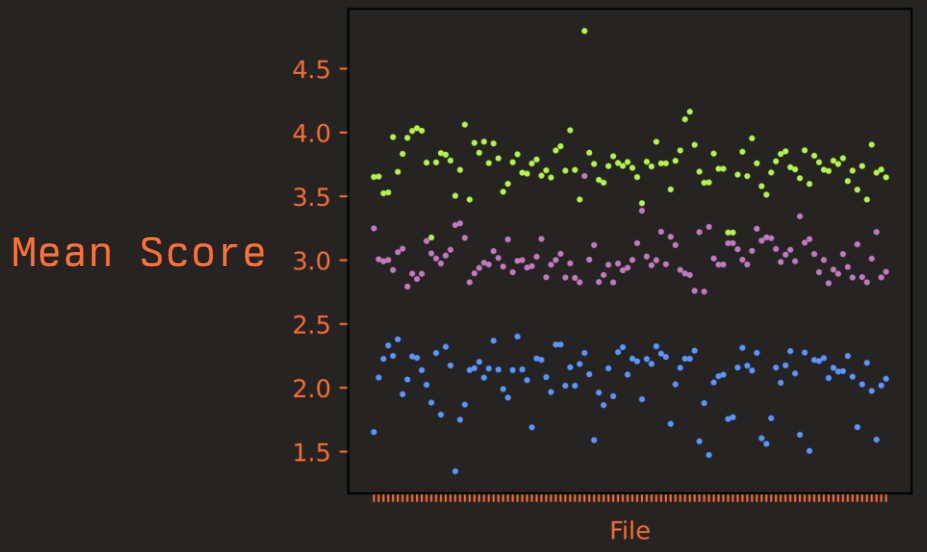
“Standard” Metric

Type considered “correct” if its C representation (with names normalized) is identical to the source type

Tool	F1 Score Coreutils	F1 Score SPEC CPU
TRex	**0.35**	**0.40**
Ghidra	0.16	0.21
ReSym	0.27	0.23

Reverse-Engineer Focused Scoring

Score assigned to type based on how close it is, prioritized based on possible frustration if incorrect



TRex: Practical Type Reconstruction for Binary Code

> Type Recovery is fundamentally impossible; focus on Type Reconstruction instead

> Reconstruction benefits from Structural Types which more precisely capture program behaviors

> Even in our increasingly machine-learning based world, deductive techniques have high value, especially when correctness matters

Comments / Questions / Tyrannosaurus Rex Memes: jayb@microsoft.com

https://github.com/secure-foundations/trex