Augmenting Decompiler Output with Learned Variable Names and Types

USENIX Security Symposium

Qibin Chen, Jeremy Lacomis, Edward J. Schwartz, Claire Le Goues, Graham Neubig, and Bogdan Vasilescu

Variable Naming is Essential

```
 1| void read(char *a1) {
 2|   int v1 = 0;
 3|   char v2;
 4|   while (1) {
 5|     v2 = getchar();
 6|     if (v2 == EOF || v2 == '\n') {
 7|       a1[v1] = '\0';
 8|       break;
 9|     } else {
10|       a1[v1] = v2;
11|     }
12|     v1++;
13|   }
14| }
```

Variable Naming is Essential

```
1| void read(char *out_buf) {
2|   int loc = 0;
3|   char cur;
4|   while (1) {
5|     cur = getchar();
6|     if (cur == EOF || cur == '\n') {
7|       out_buf[loc] = '\0';
8|       break;
9|     } else {
10|       out_buf[loc] = cur;
11|     }
12|     loc++;
13|   }
14| }
```

Variable Naming is Essential

```
1| void read(char *banana) {
2|   int str = 0;
3|   char moo;
4|   while (1) {
5|     moo = getchar();
6|     if (moo == EOF || moo == '\n') {
7|       banana[str] = '\0';
8|       break;
9|     } else {
10|       banana[str] = moo;
11|     }
12|     str++;
13|   }
14| }
```

DIRE

```
1| file *f_open(char **filename, char *mode, int create) {
2|   int fd;
3|   if (!create)
4|     return fopen(*filename, mode);
5|   if (*mode != 119)
6|     assert_fail("fopen");
7|   fd = open(*filename, 577, 384);
8|   if (fd >= 0)
9|     return reopen(fd, mode);
10|   else
11|     return 0;
12| }
```

Hex-Rays	Developer
a1	filename
a2	mode
a3	is_private

Types Are Also Useful

```
double func(int *a1, int *a2) {
  double v1, v2;
  v1 = pow((*a1 - *a2), 2);
  v2 = pow((a1[1] - a2[1]), 2);
  return sqrt(v1 + v2);
}
```

Types Are Also Useful

```
double func(int *a1, int *a2) {
  double v1, v2;
  v1 = pow((*a1 - *a2), 2);
  v2 = pow((a1[1] - a2[1]), 2);
  return sqrt(v1 + v2);
}
```

```
typedef struct {
  int x;
  int y;
} point;

double func(point *a1, point *a2) {
  double v1, v2;
  v1 = pow((a1->x - a2->x), 2);
  v2 = pow((a1->y - a2->y), 2);
  return sqrt(v1 + v2);
}
```

Predicting Types should be easier than names!

```
typedef struct {
  int x;
  int y;
} point;

sizeof(char);   // 1
sizeof(int);    // 4
sizeof(float);  // 4
sizeof(point);  // 8
sizeof(int[2]); // 8
```

Constraints Experiment

Constraints Experiment

- Mine GitHub, creating the same dataset as for DIRE

Constraints Experiment

- Mine GitHub, creating the same dataset as for DIRE
- Also maintain a database of types including their substructure

Constraints Experiment

- Mine GitHub, creating the same dataset as for DIRE
- Also maintain a database of types including their substructure
- Train a model to predict a type based on its context and constrain by size

Constraints Experiment

- Mine GitHub, creating the same dataset as for DIRE
- Also maintain a database of types including their substructure
- Train a model to predict a type based on its context and constrain by size
- Results: Quite Bad!

The Problem: Padding Bytes

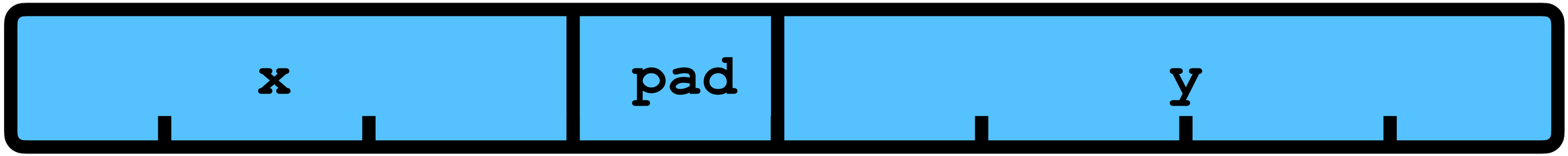
Original

```
void fun() {
  char x[3];
  int y;
  // ...
}
```

Decompiled

```
void fun() {
  char x[4];
  int y;
  // ...
}
```

Stack



The Problem: Padding Bytes

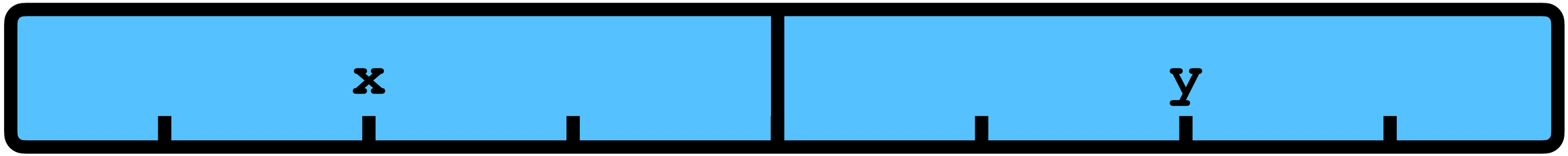
Original

```
void fun() {
  char x[3];
  int y;
  // ...
}
```

Decompiled

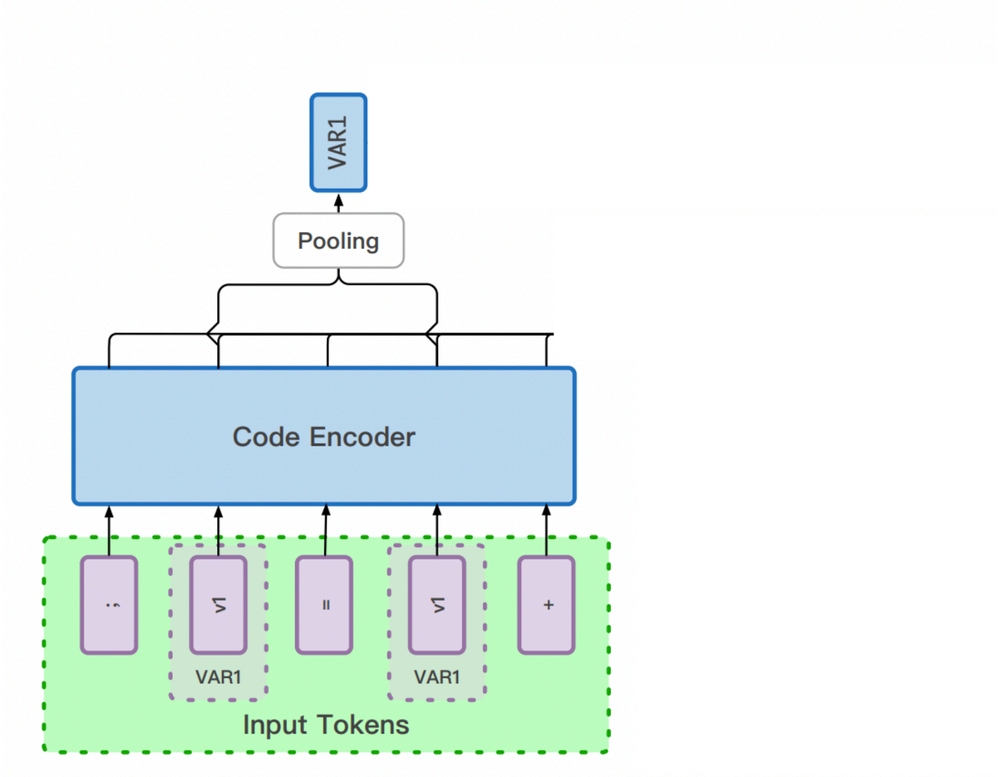
```
void fun() {
  char x[4];
  int y;
  // ...
}
```

Stack

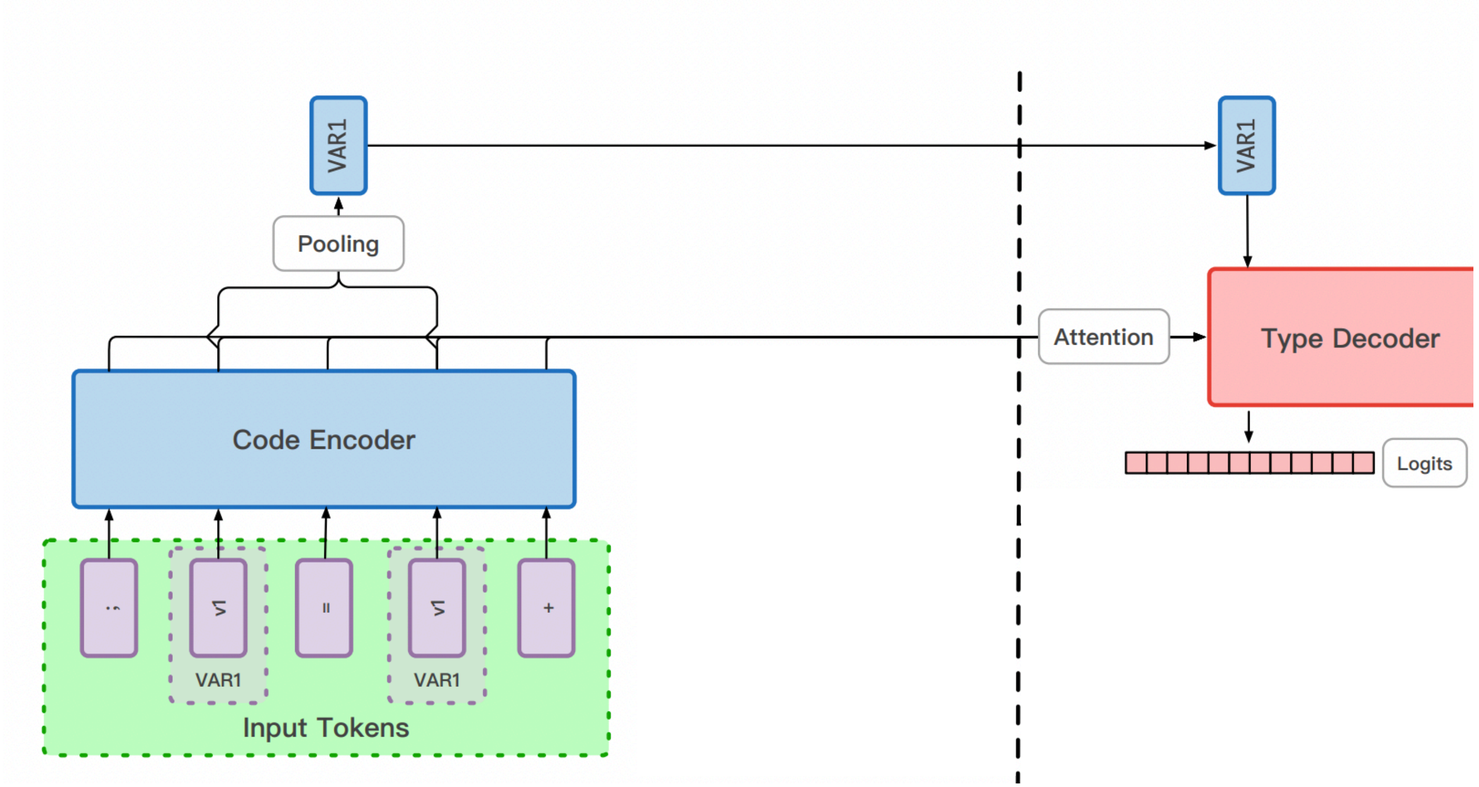


DecompIled variable ReTYper (DIRTY) Architecture

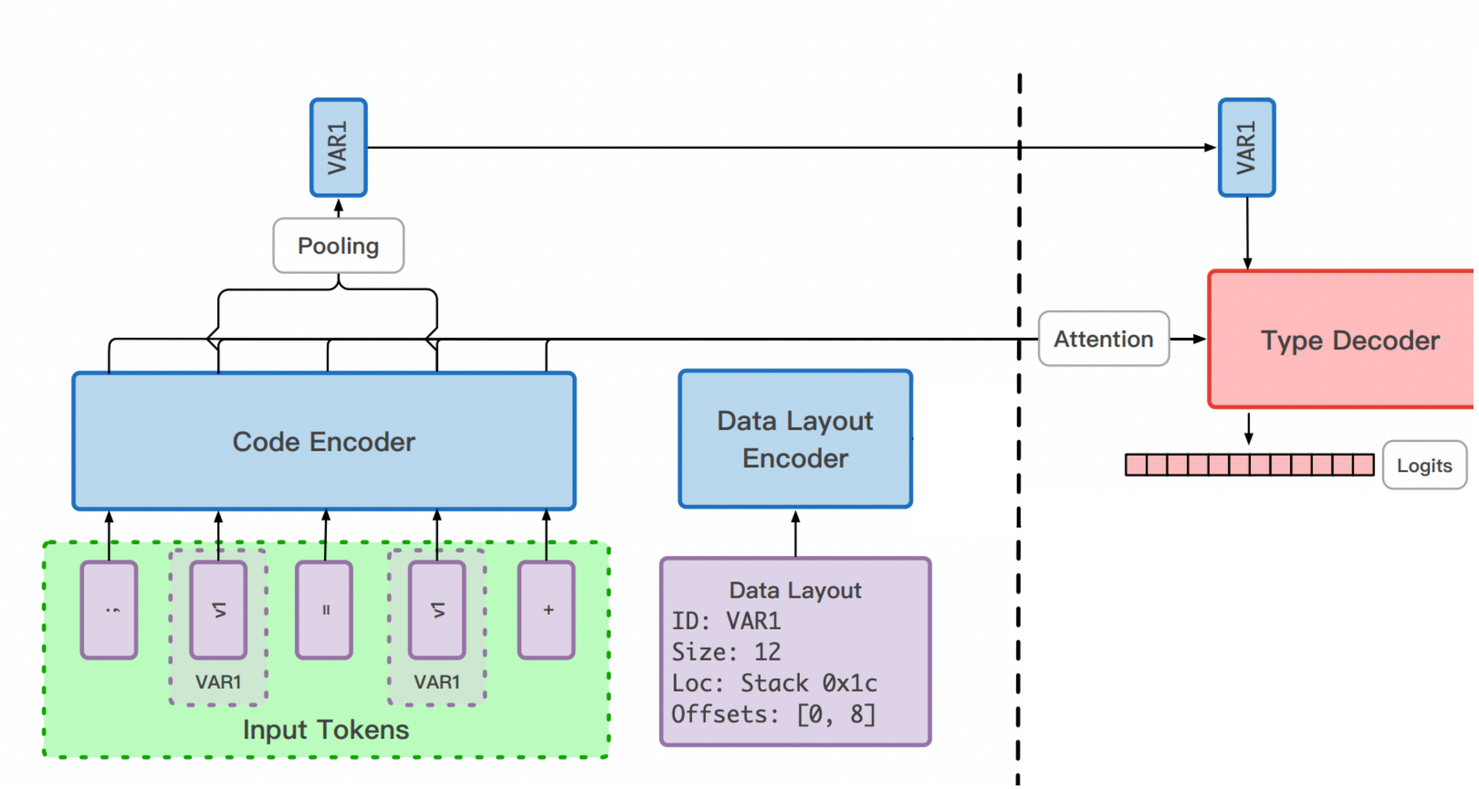
DecompIled variable ReTYper (DIRTY) Architecture



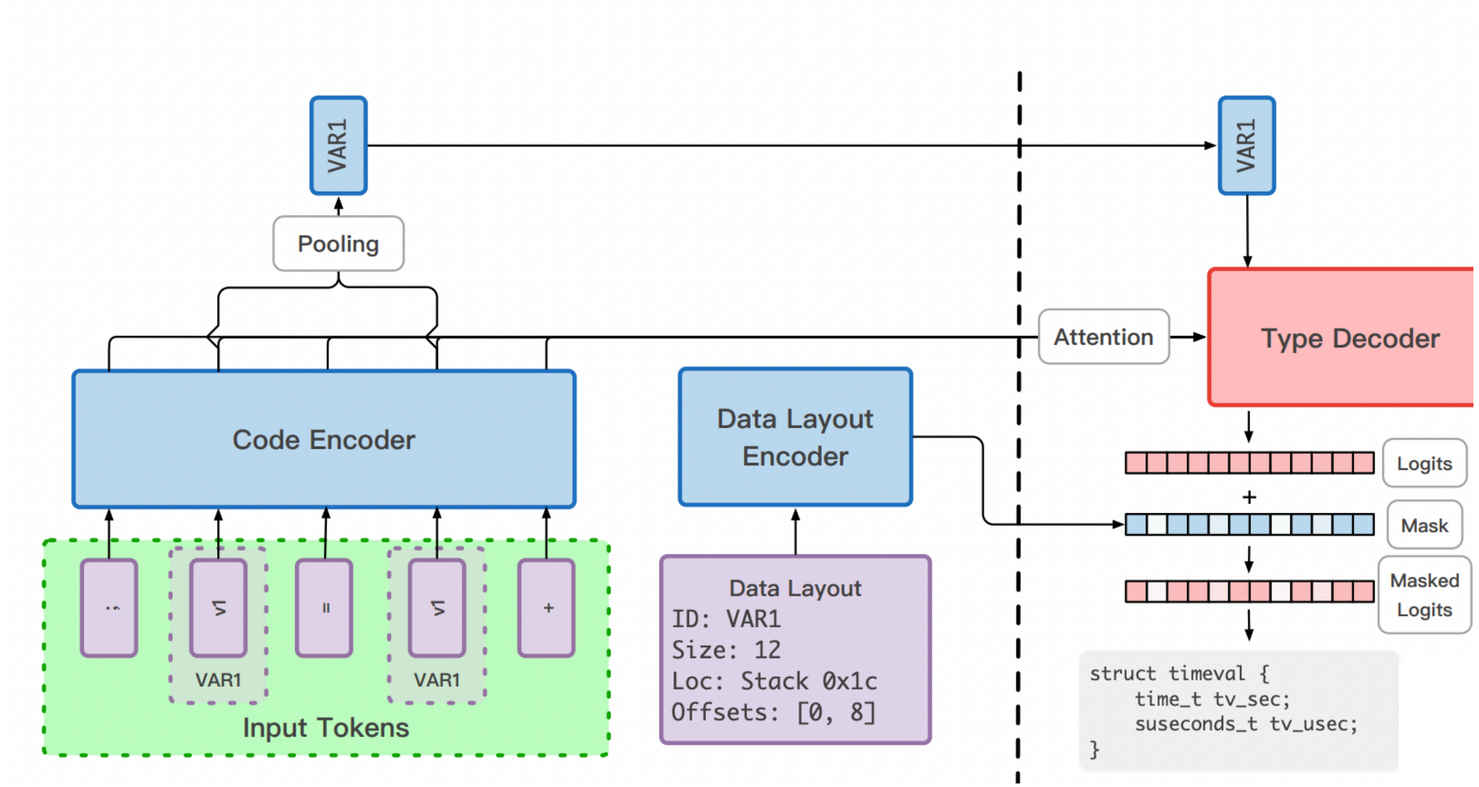
DecompIled variable ReTYper (DIRTY) Architecture



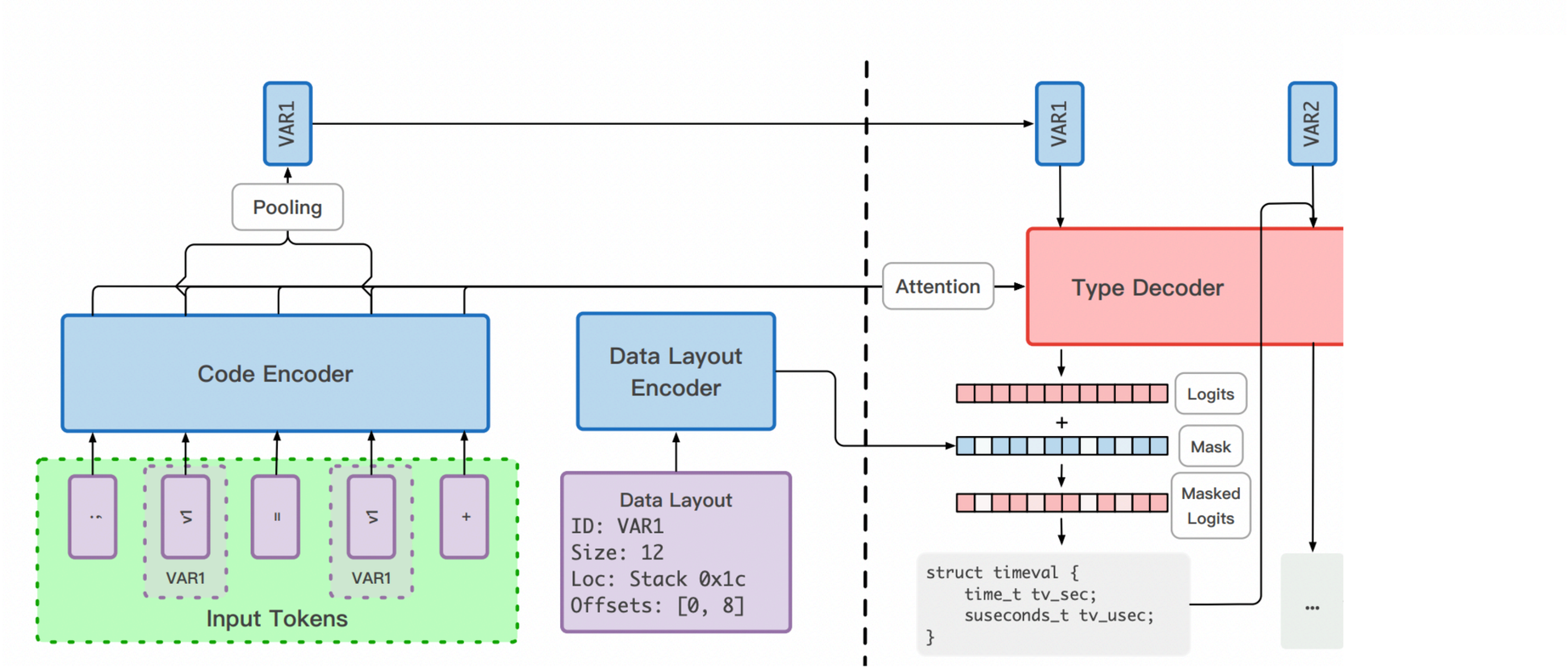
DecompIled variable ReTYper (DIRTY) Architecture



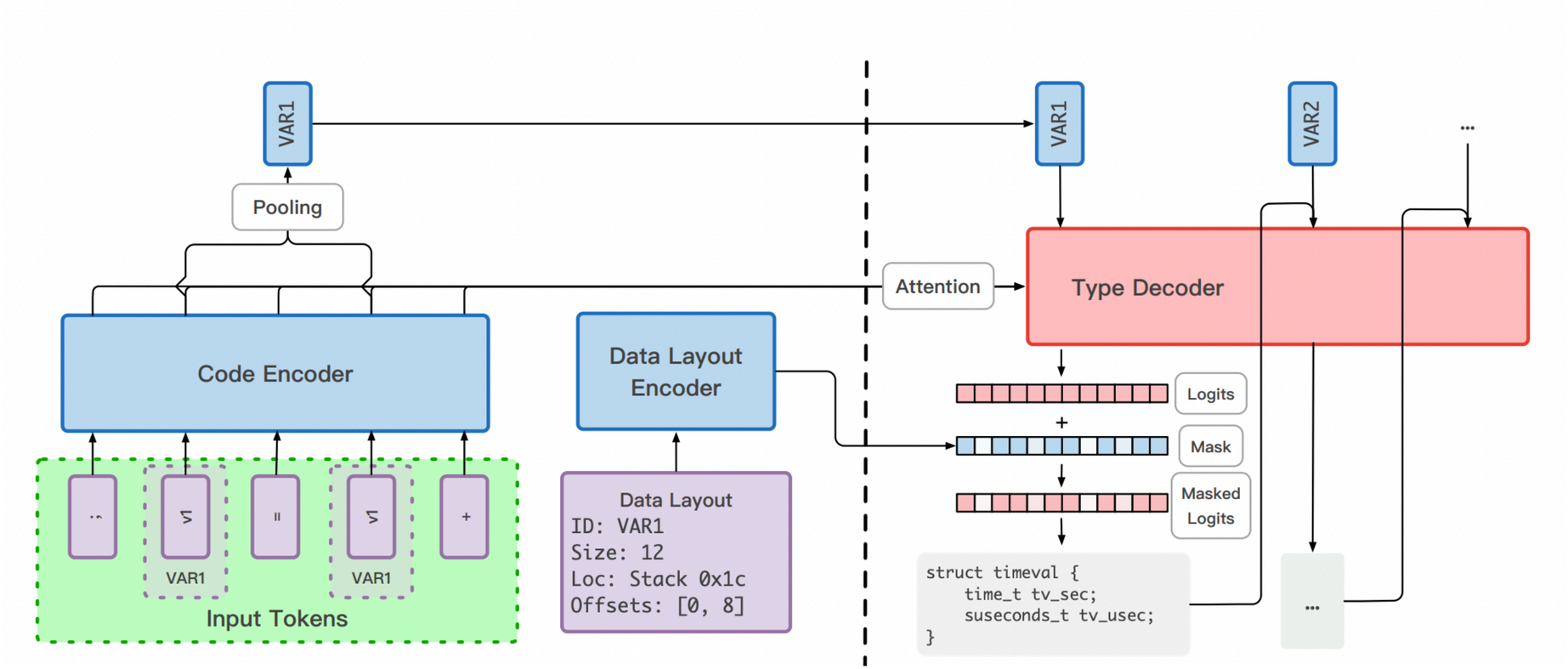
DecompIled variable ReTYper (DIRTY) Architecture



Decompiled variable ReTYper (DIRTY) Architecture



Decompiled variable ReTYper (DIRTY) Architecture



Evaluation

- DIRTY was evaluated on the Dataset for Idiomatic ReTyping (DIRT)

Evaluation

- DIRTY was evaluated on the Dataset for Idiomatic ReTyping (DIRT)
- Baselines were Hex-Rays and Frequency by Size

Evaluation

- DIRTY was evaluated on the Dataset for Idiomatic ReTyping (DIRT)
- Baselines were Hex-Rays and Frequency by Size
- Accuracy: the percentage of types that exactly match the ground truth, including data layout, type name, and substructure, where applicable

Evaluation

- DIRTY was evaluated on the Dataset for Idiomatic ReTyping (DIRT)
- Baselines were Hex-Rays and Frequency by Size
- Accuracy: the percentage of types that exactly match the ground truth, including data layout, type name, and substructure, where applicable

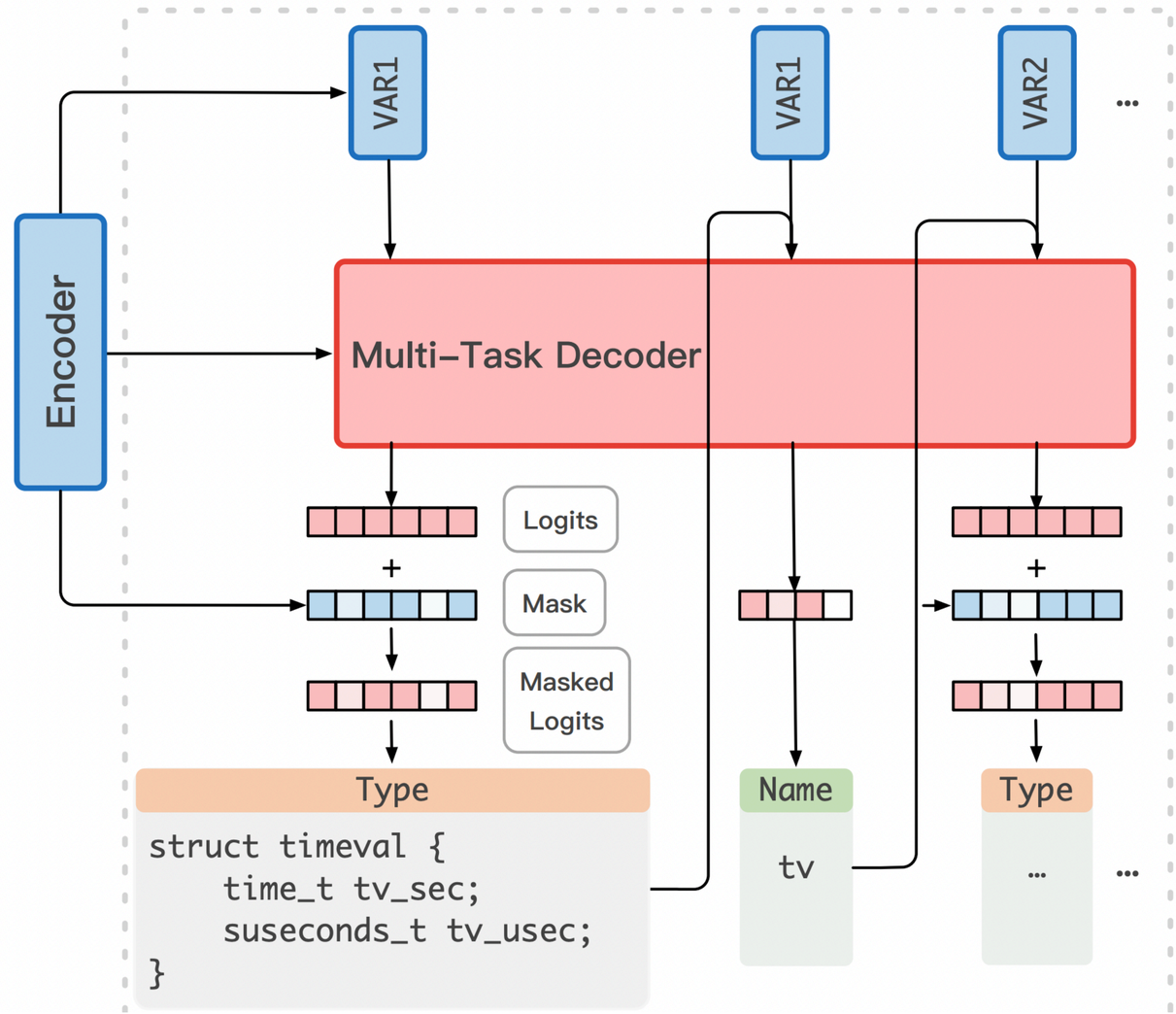
Method	All	Structures
Freq by Size	23.6	9.7
Hex-Rays	37.9	28.7
DIRTY	75.8	68.6

Compiler Optimization Performance

Percent accuracy on 101 GNU Coreutils Programs

-O0	-O1	-O2	-O3
48.20	46.01	46.04	46.00

Predicting types and names simultaneously



Evaluation

- DIRTY compared with DIRE

Evaluation

- DIRTY compared with DIRE
- Used the dataset from the original DIRE paper and also DIRT

Evaluation

- DIRTY compared with DIRE
- Used the dataset from the original DIRE paper and also DIRT
- Accuracy is the percentage of names that exactly match the developer's choice.

Evaluation

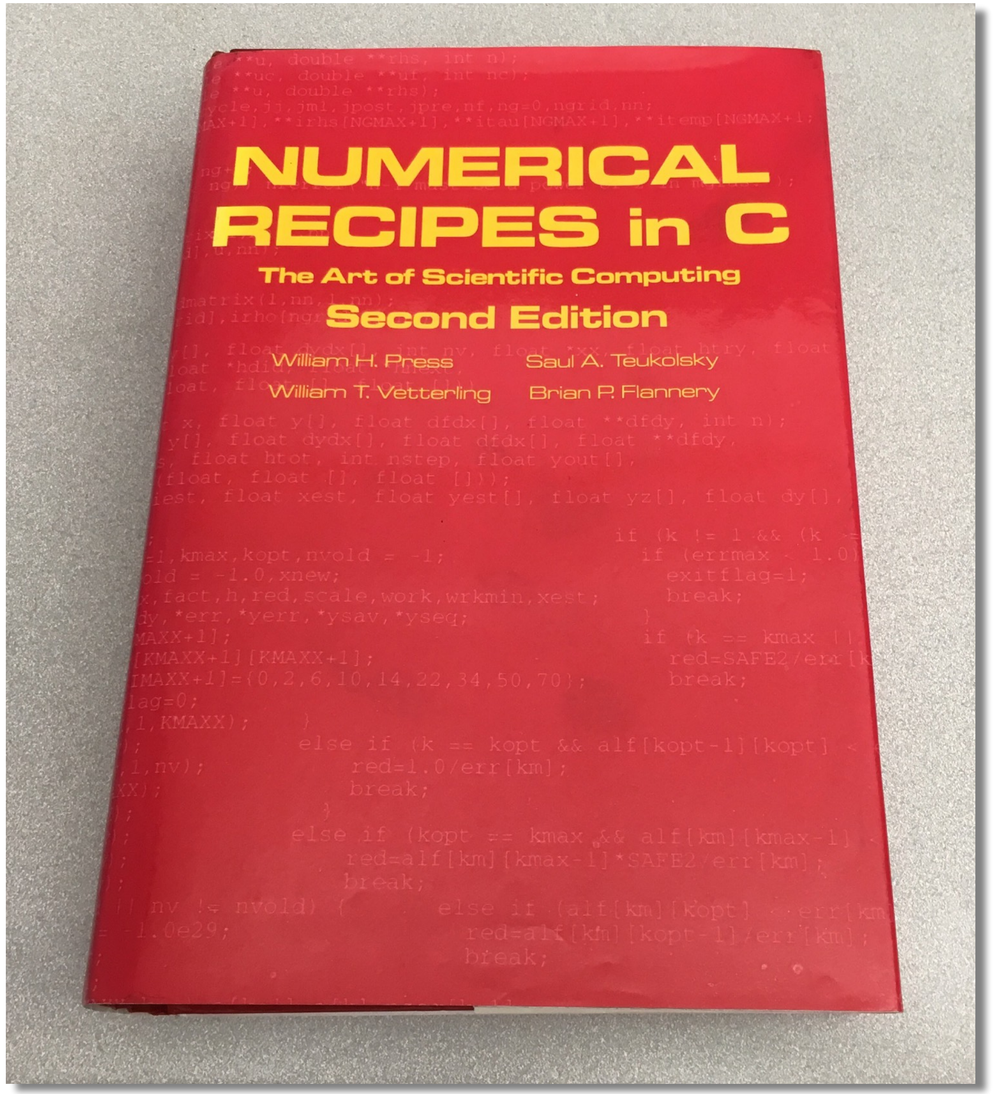
- DIRTY compared with DIRE
- Used the dataset from the original DIRE paper and also DIRT
- Accuracy is the percentage of names that exactly match the developer's choice.

Method	DIRE Data	DIRT
DIRE	72.8	57.5
DIRTY	81.4	66.4

```
int find_unused_picture(int a1, int a2, int a3) {
  int i, j, v1;
  if (a3) {
    for (i = <NUM>;; ++i) {
      if (i > <NUM>)
        goto LABEL_13;
      if (!*(*(<NUM> * i + a2) + <NUM>))
        break;
    }
    v1 = i;
  } else {
    for (j = <NUM>;; ++j) {
      if (j > <NUM>) {
      LABEL_13:
        av_log(a1, <NUM>, <STR>);
//...
```

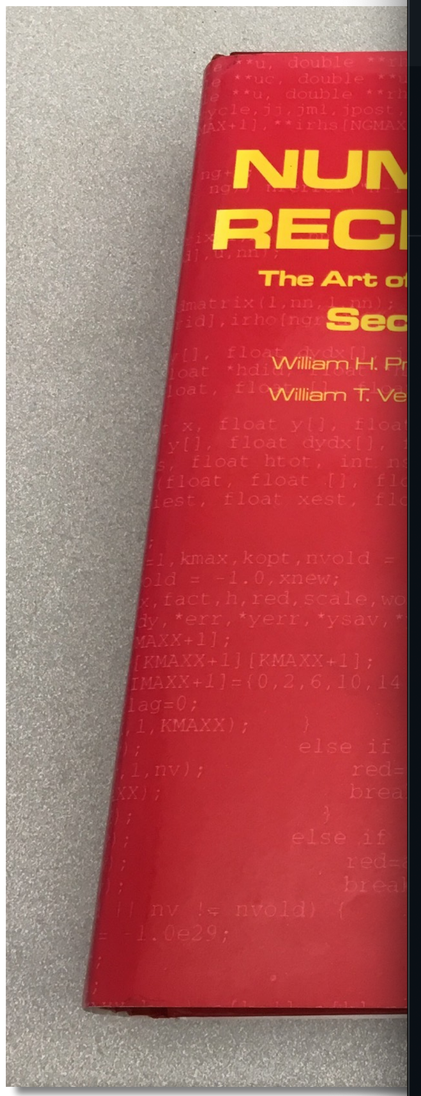
ID	Developer	DIRTY
a1	AVCodecContext *avctx	MpegEncContext *s
a2	Picture *picture	Picture *pic
a3	int shared	int shared
v1	int result	int result

BONUS SLIDES




```
void scrsho(float (*fx)(float))
For interactive CRT terminal use.  Produce a crude graph of th
for interval x1,x2.  Query for another plot until the user sign
{
    int jz,j,i;
    float ysml,ybig,x2,x1,x,dyj,dx,y[ISCR+1];
    char scr[ISCR+1][JSCR+1];

    for (;;) {
        printf("\nEnter x1 x2 (x1=x2 to stop):\n");
        scanf("%f %f",&x1,&x2);
        if (x1 == x2) break;
        for (j=1;j<=JSCR;j++)                          Fill ver
            scr[1][j]=scr[ISCR][j]=YY;
        for (i=2;i<=(ISCR-1);i++) {
            scr[i][1]=scr[i][JSCR]=XX;                 Fill top
            for (j=2;j<=(JSCR-1);j++)                  Fill int
                scr[i][j]=BLANK;
        }
        dx=(x2-x1)/(ISCR-1);
```



books/scrsho.cpp at master · b

github.com/burakbayramli/books/blob/master/NumericalRecipes2/recipes/scrsho.cpp

Search or jump to... / Pulls Issues Marketplace Explore

burakbayramli / books

Watch Fork Star 148

Code Issues Pull requests

master books / NumericalRecipes2 / recipes / scrsho.cpp

BB Initial commit — Latest commit d32c78b on Jan 3, 2017 — History

0 contributors

55 lines (53 sloc) | 1.25 KB — Raw Blame

```
#include <string>
#include <iostream>
#include <iomanip>
#include "nr.h"
using namespace std;

void NR::scrsho(DP fx(const DP))
{
        const int ISCR=60, JSCR=21;
        const char BLANK=' ', ZERO='-', YY='l', XX='-', FF='x';
        int jz,j,i;
        DP ysml,ybig,x2,x1,x,dyj,dx;
        Vec_DP y(ISCR);
        string scr[JSCR];

        for (;;) {
                cout << endl << "Enter x1 x2 (x1=x2 to stop):" << endl;
                cin >> x1 >> x2;
                if (x1 == x2) break;
                scr[0]=YY;
```