



# Damo

## A language for symbolic functions

Ian Covert

Project Manager

Abhiroop Gangopadhyay

Language Guru

Srihari Devaraj

System Architect

Alan Gou

Testing



The background of the slide is a complex network diagram. It consists of numerous small, light gray circular nodes connected by thin, light gray lines. Some nodes are highlighted with a darker gray or blue color, and some are enclosed in a dashed circular border. The overall pattern is dense and interconnected, suggesting a network or data structure.

# I. Introduction


## Why...

# CORE FEATURES

- Scripting language
- Symbolic expressions
- Standard library for symbolic function evaluation, automatic differentiation

# MOTIVATION

A decorative network diagram in the top right corner, consisting of various sized circles (nodes) connected by thin lines (edges). Some nodes are highlighted with a blue glow, and the overall structure is sparse and interconnected.

- Ease of development for applications requiring automatic differentiation
  - Useful for many kinds of machine learning, such as SGD algorithm for neural networks
  - Historical note:
    - [Damo](#) was child of [Theano](#) – a popular Python deep learning library
- 
- A decorative network diagram in the bottom left corner, similar to the one in the top right, featuring nodes and connecting lines, with some nodes highlighted in blue.

The background of the slide is a light gray network of interconnected nodes and lines, resembling a social or organizational graph. The nodes are represented by small circles, some solid and some hollow, connected by thin lines. The overall pattern is dense and covers the entire background.

## II. Project Management

### Responsibilities, lessons learned

# EVERYONE BECOMES A DEVELOPER



Abhiroop

Built SAST, semantic checking and tests



Ian

Implemented parser, standard library and tests



Hari

Implemented codegen and tests



Alan

Took the worst classes of his life this semester

# LESSONS LEARNED

- Iterative development makes life easier
- Specialization is helpful, but dangerous
- Realistic deadlines are necessary
- Never assume that something works



# III. The Damo Language

## Speaking our dialect



# SYNTAX BASICS

```
// Single line comment
```

```
/*  
    Multiline  
    comment  
*/
```

```
// VARIABLE DECLARATION
```

```
int i;  
int j = 1;
```

```
/*  
OPERATORS  
+ - * / ^ _ %  
and or not  
< > <= >= == !=
```

```
TYPES
```

```
int  
num  
bool  
string  
symbol  
*/
```

# SIMPLE PROGRAMS

```
print("Hello world");

// FUNCTION DECLARATION
def sayHello(string name) : void {
    print("Hello, ");
    print(name);
}

sayHello("Stephen");
```

```
num a;
num b;
num c;

a = 1;
b = 2.0;
c = a * b;

print_num(c);
```

# CONTROL FLOW

```
// C-like loops

int i;

print("Going up");
for (i = 0; i < 10; i = i + 1){
    print_int(i);
}

print("Going down");
while (i > 0){
    print_int(i);
    i = i - 1;
}
```

```
// C-like if-elseif-else statements

if (i < 0){
    print("i less than 0");
}
elseif (i < 10){
    print("i less than 10");
}
else {
    print("i greater than 10");
}
```

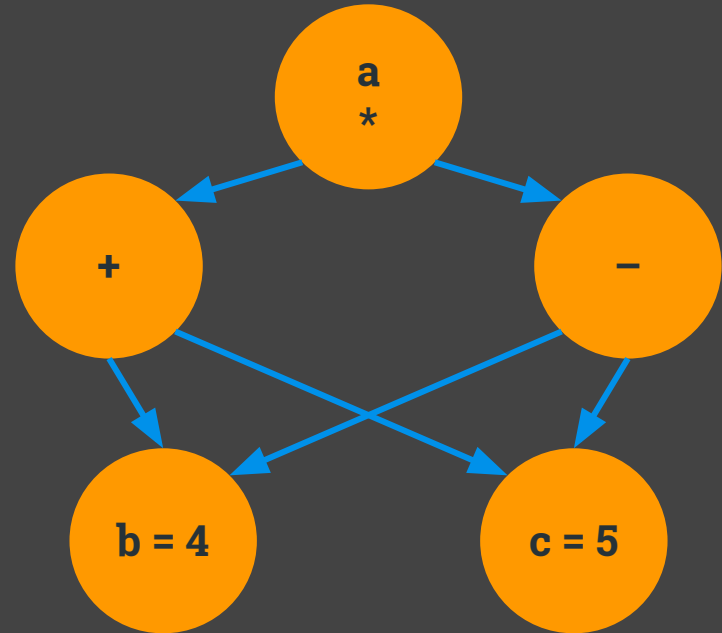
# SYMBOLIC EXPRESSIONS

```
// Declare symbols
symbol a;
symbol b;
symbol c;

// Set symbolic expression
a = b + c;
a = a * (b - c);

// Set symbols to constant values
b = 4;
c = 5;
```

```
// DEPENDENCY GRAPH
```



# THE STANDARD LIBRARY

```
// Function evaluation
symbol a; symbol b; symbol c;

a = b * c;
b = 4;
c = 5;

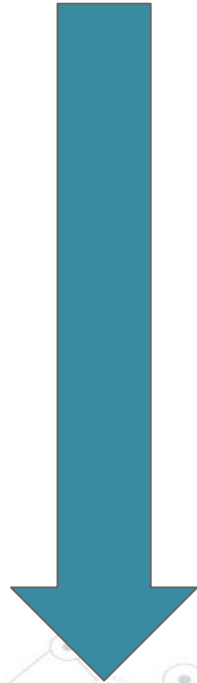
num result = eval(a);
num deriv = partialDerivative(a, b);
```

The background of the slide is a light gray network diagram. It consists of numerous small circular nodes, some of which are highlighted with a darker gray or blue color. These nodes are interconnected by thin, light gray lines, creating a complex web-like structure that fills the entire frame. The overall aesthetic is clean and technical, typical of a presentation on networking or data science.

# IV. How it works

## Implementation details


# OUR COMPILATION PIPELINE



<b>Source code</b>	Program written in Damo
<b>Linked with stdlib</b>	Standard library prepended
<b>Symbols</b>	Scanner
<b>AST</b>	Parser
<b>SAST</b>	Semantic checking
<b>LLVM</b>	Codegen
<b>Executable</b>	C compiler: links with C code

# ABSTRACT SYNTAX TREE

A decorative network diagram in the top right corner, consisting of various sized circles (nodes) connected by thin lines (edges). Some nodes are solid grey, while others are hollow white with a grey outline. The connections form a complex, branching structure.

- A Damo program is a list of variable declarations, function declarations, and statements
  - Arbitrary ordering
  - Semantic checking verifies proper usage of variables and functions
- 
- A decorative network diagram in the bottom left corner, similar to the one in the top right, featuring a mix of solid and hollow nodes connected by lines.



# HEAP ALLOCATED SYMBOLS

- Invoke C functions to allocate heap memory

```
symbol_malloc = Lllvm.declare_function "createSymbol" (Lllvm.function_type
symbol_t [| |] the_module
...
A.Symbol -> let global_variable = L.build_call symbol_malloc [| |] "symbolmal"
builder in ignore(L.build_store global_variable s_v builder);
```

# THE SYMBOL STRUCT

- Underlying C struct represents symbol type

```
struct symbol {  
    symbol *left;  
    symbol *right;  
    int isConstant;  
    int isInitialized;  
    double value;  
};
```

# LINKING WITH C CODE

- Makefile builds `symbol.c`, a library we wrote to handle routines relating to symbols
  - Heap memory allocation
  - Accessor, mutator functions
- Damo executables are linked with C standard library, and `symbol.o`


The background of the slide is a light gray network of interconnected nodes and lines, resembling a social or data network. The nodes are represented by small circles, some solid and some hollow, connected by thin lines. The overall pattern is dense and covers the entire background.

# V. Testing

## It works, we promise

# UNIT AND INTEGRATION TESTS



- We tested for every feature of the Damo language
    - Operators
    - Functions
    - Global variables
    - Standard library functions
    - Etc.
- 

# THE ULTIMATE TEST

- Showing off in our demo – a big integration test

The background of the image is a light gray network graph. It consists of numerous small circular nodes, some of which are solid gray and others are hollow white with a gray outline. These nodes are interconnected by a web of thin, light gray lines, creating a complex, interconnected pattern that fills the entire frame. The overall aesthetic is clean and technical, suggesting a theme of networking, data, or technology.

Let's demo it!