SCIC

Scientific Calculation Language

Eleven Li, Yucen Sun, Zhengyuan Dong

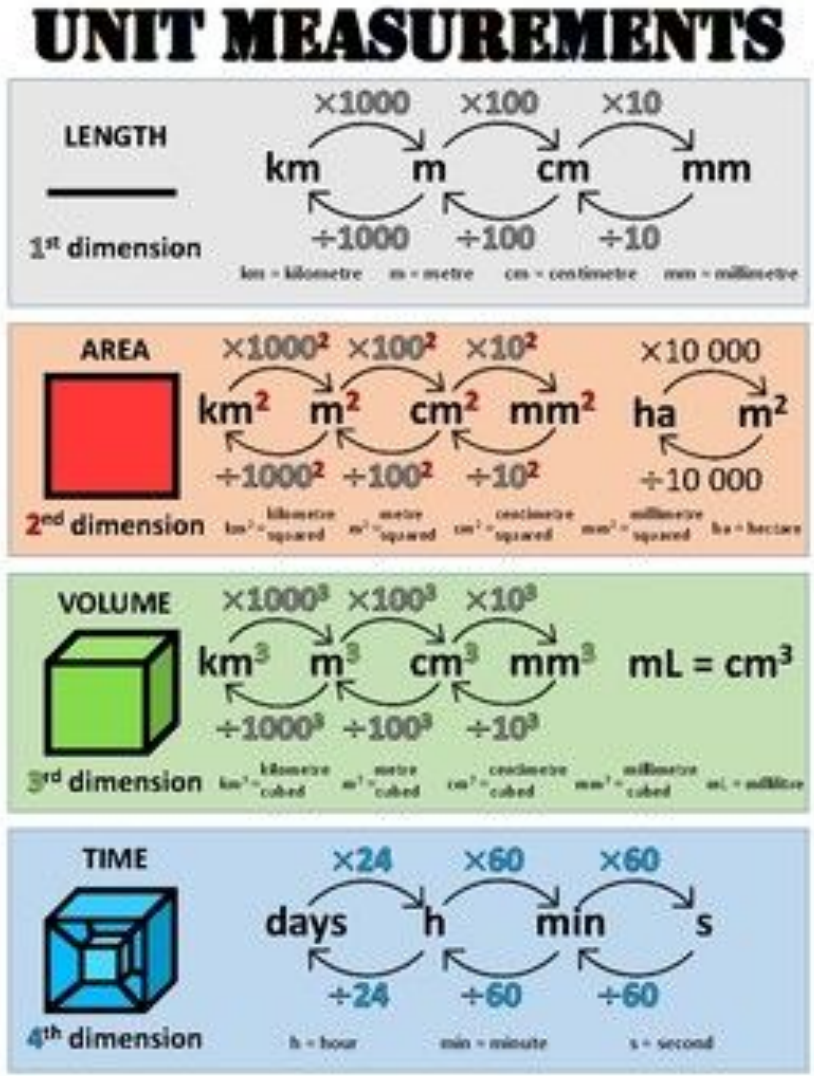
CONTENTS

1.St

- Overview
- Compiler Architecture
- Features
- Demo
- Teamwork

Motivation

- Manual unit conversion in the code can be cumbersome and error-prone
- Problem in the history:
  Loss of Mars Climate Orbiter - A catastrophic industrial accident due to unit problems
- Our goal: a general purpose language with user-friendly unit features



Language Overview

Static

Statically typed
Statically scoped
Data type and unit explicitly specified

01

C-like Syntax

Syntactically mimic C
with new data type and unit syntax

04

Unit

03

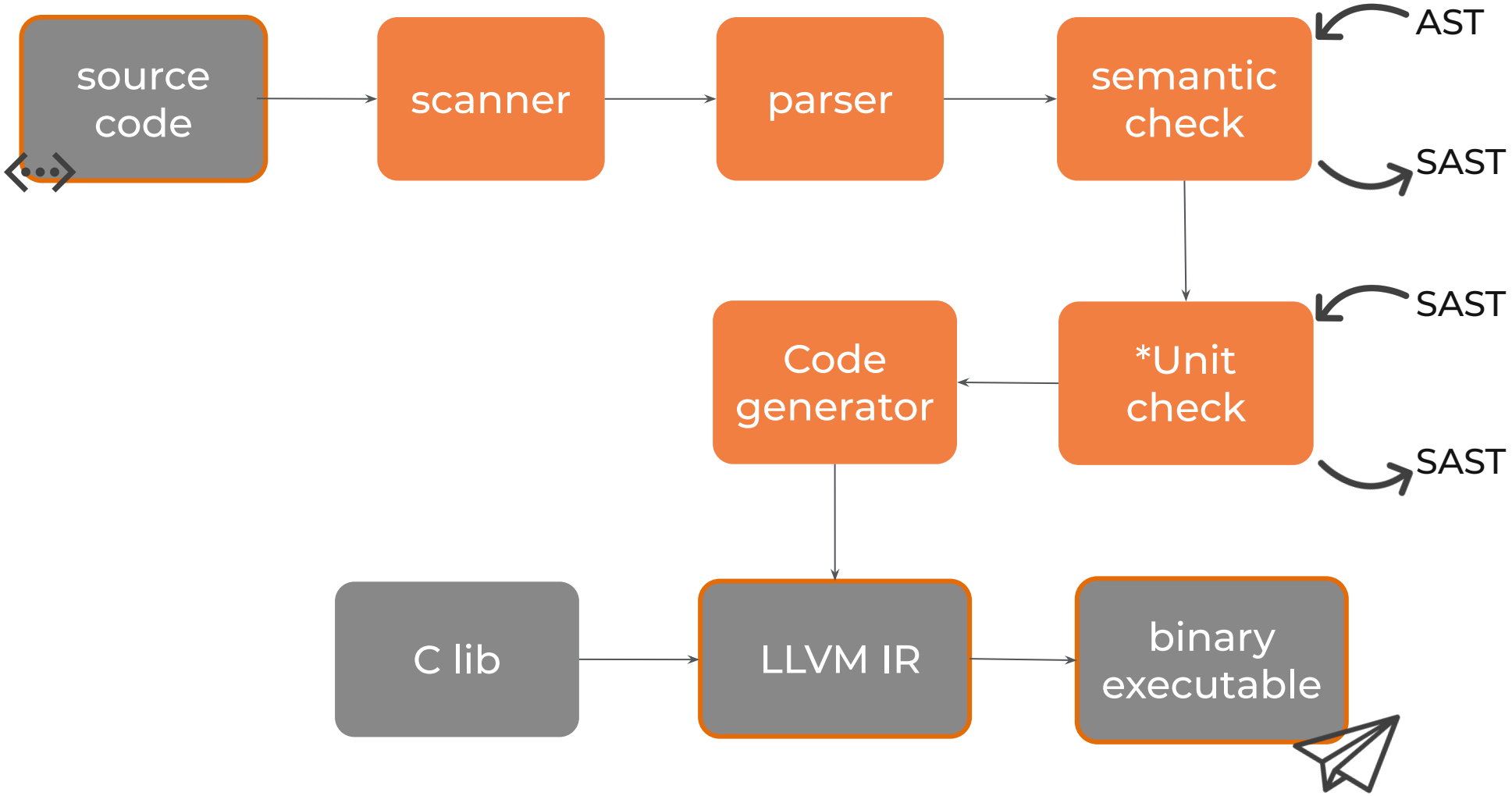
Supports variables with units,
Build-in base and user-defined units,
automatic unit conversion,
supported units: m, cm, s, kg, m/s ...

02 **Types**

int, float, boolean, string, void
int array, float array


2nd
Overview
Compiler Architecture
Features
Demo
Teamwork

Compiler Architecture



Focus on the normal semantic check like type check and expr validation. Walk over AST and generate SAST.

*An additional layer dedicated to check units for variables, exprs and functions;
modify sexprs to do unit conversion, no change in type


3.rd
Overview
Compiler Architecture
Features
Demo
Q&A
Teamwork

Syntax & Grammer

A SCIC program is composed of:

- Unit declarations
- Global variable declarations
- Function declarations

Declared units are global

Inside function declarations, we can define local variables (must initialize).

Static type and static unit:

Must specify legal type and unit (or no-unit) upon variable/function declaration.

Only float and float array has unit, other types (bool, string, etc) no-unit.

```
float '{cm} a_has_unit = 1.2;
float b_no_unit = 1.3;
```

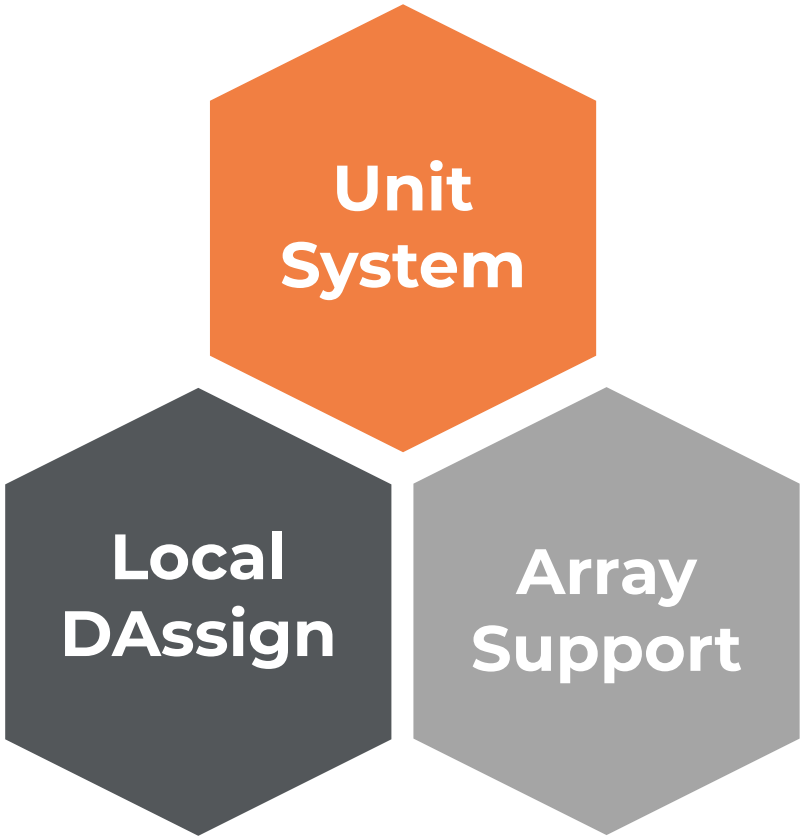
Support **float array with static unit**:

Every element in one array has the same unit.

Support for, while, if-else stmts.

```
int func main() {
    int i = 0;
    float[] '{m} dx = [2.3, 4.5, 3.4, 0.7];
    float[] '{s} dt = [0.5, 0.2, 1.7, 0.5];
    float[] '{m/s} res = [0.0, 0.0, 0.0, 0.0];
    for (; i < 4; i = i + 1) {
        res[i] = dx[i] / dt[i];
        printf(res[i]);
    }
    return 0;
}
```

Key Features



Unit System

|'{km} = 0.001 '{m}|;

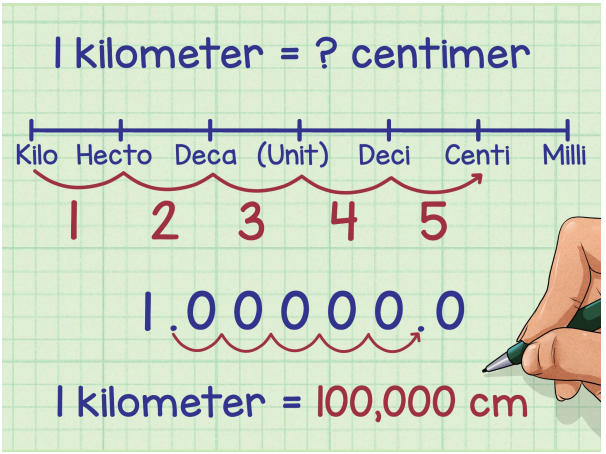
float '{cm} x = 30.0;

Local Declare-and-assign

Flexible local variables

Array Support:

Support array and operations



Unit System: Unit declaration

Base units: S.I. Unit

S.I. Unit	
Name	**Symbol**
kilogram	kg
metre	m
second	s
ampere	A
kelvin	K
mole	mol
candela	cd

Non-base units:
centimeter: cm
Others leave to the users

User defined unit is a scaling of an existing unit.
The scale is a float number.

```
|'{us} = 1.0E+6 '{s}|;
float '{us} ua;
```

```
|'{mm} = 1000.0 '{m}|;
|'{km} = 0.001 '{m}|;
```

Derived unit (a simple expr of existing units) can be used directly.

```
|'{g} = 1000.0 '{kg}|;

float '{m*g} x;

int func main() {
    x = 9.98;
    float '{cm*kg} y = x;
    printf(y);
    return 0;
}
```

The conversion to target unit is automatic!
(Here prints 0.998)

Unit System: Unit declaration

- Only float and float array have units. Unit default to none for other types (bool, int, string ...). If a float is defined without '{unit}, the compiler interpret it as no-unit.
- Same rule applies to function formals and function return value's unit.
- All element in a float array have the same unit. If an element is accessed, the element keeps its unit.

```
float '{km} func foo(float '{m} x) {
    float '{km} y = x;
    return y;
}
```

```
|'{ms} = 1000.0 '{s}|;

int func main() {
    float[] '{cm} x = [1.1, 2.2, 3.3];
    float '{m} y = 4.4;
    y = x[1];
    printf(y); /*2.2cm = 0.022m, prints 0.022*/
    return 0;
}
```

Keep a Set for base_units and a Map for units. Manage the set and map in unitcheck.ml

Unit System: Unit conversion

```
1   /* test local var declaration with unit */
2
3   float '{cm} y;
4   int func main(){
5       float '{m} z = 0.05;
6       printf(z); /* 0.05 m*/
7       y = z;     /* auto-conversion cm<-m*/
8       printf(y); /* 5 cm*/
9       printf(z); /* 0.05 m<- z unchanged*/
10      return 0;
11  }
```

Implementation:

Auto-conversion between exprs:

Unit propagation

When unichecking exprs, recursively

- check and determines the unit for current expr
- calculate the scale (a float number)
- apply the scale by wrapping the expr with a multiplication expr

---

Unit System: Unit conversion

At assign, function arguments and function return, the conversion can be seen explicitly.

```
1   |'{km} = 0.001 '{m}|;
2
3   /* parameters can be auto-converted
4       when paased into function */
5   float '{km} func foo(float '{m} x) {
6       printf(x); /* 30 (m) */
7       float '{m} y = x;
8       return y;    /* at return: convert to return unit km */
9   }
10
11  int func main() {
12      float '{cm} tmp = 3000.0;
13      printf(foo(tmp));  /* 0.03 (km)*/
14      return 0;
15  }
```

Some of the rules in unicheck:

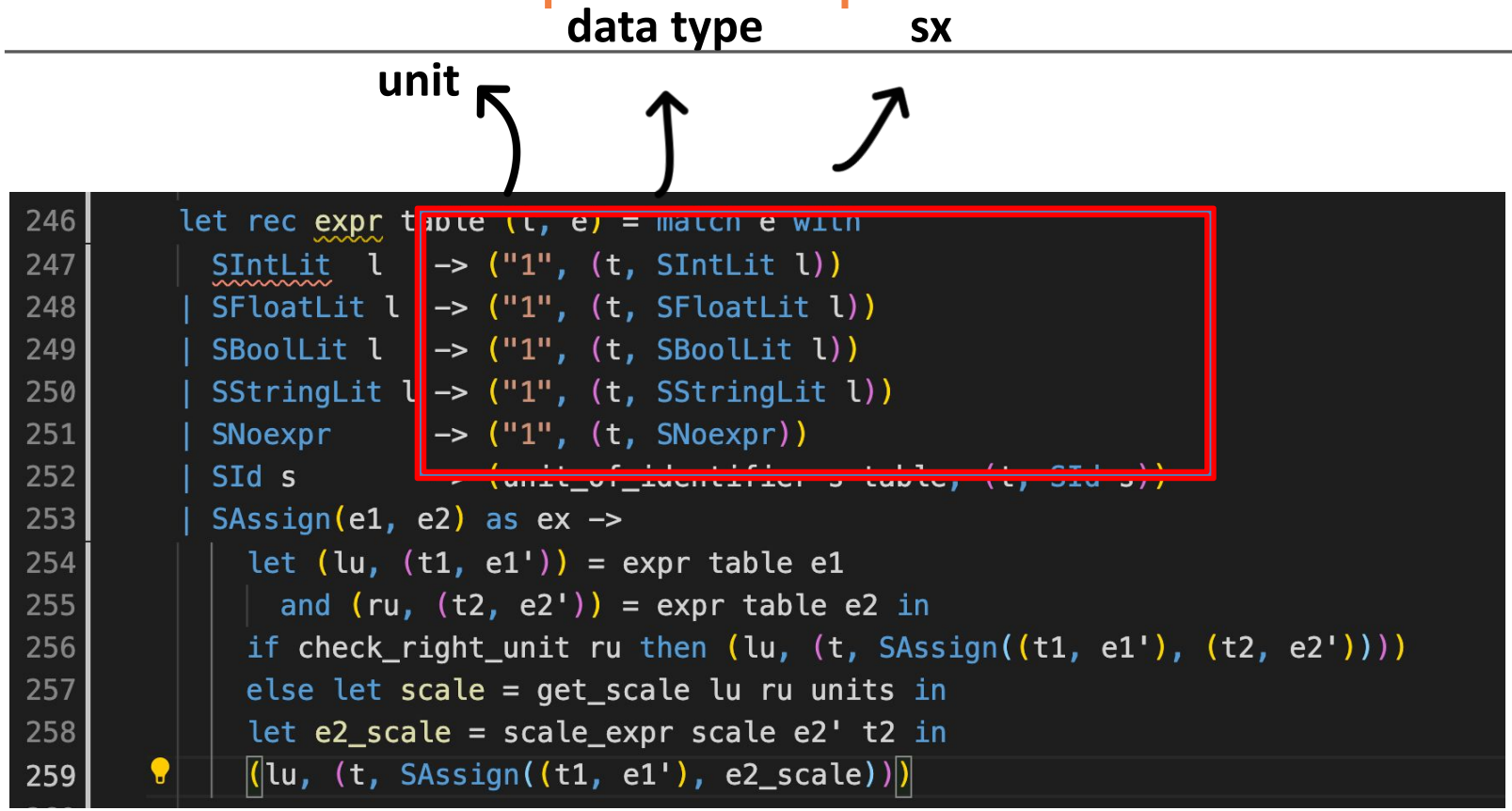
eu -> u1 + u2:

- if u1 = no-unit,
  - follow u2;
- else if both have units,
  - eu = u1; scale u2;
  - if u2 cannot convert to u1,
  - raise error

eu -> u1 *u2:

- if u1 or u2 is no-unit:
  - follow the other;
- else:
  - eu = u1 * u2

Unit Checker Input & Ouput



Unitcheck.ml layer takes in SAST, generate Unit-checked SAST, put into codegen

Unit Checker Data Structures

```
let check (udecls, globals, functions) =
  (* base unit set - static *)
  let base_units =
    List.fold_right SS.add ["m"; "s"; "kg"; "A"; "K"; "mol"; "cd"; "1";] SS.empty
  in

  (* unit maping key: non-base unit, value: (base unit, scale)*)
  let units =
    StringMap.add "cm" ("m", 100.0) StringMap.empty
  in
```

basic units set

unit conversion table

```
68      let add_unit table (u1, u2, c)  =
69        match SS.find_opt u2 base_units with
70          Some bu -> StringMap.add u1 (u2, float_of_string c) table
71        | None -> match StringMap.find_opt u2 table with
72                    Some (bu, c2) -> StringMap.add u1 (bu, c2 *. (float_of_string c)) table
73                  | None -> raise (Failure ("The reference unit not existed " ^ u2))
74      in
75
76      let add_unit_decls (unit_decls: unit_decl list) table =
77        ignore(check_udecls "new unit" unit_decls);
78        List.fold_left add_unit table unit_decls
79      in
80
81      let units = add_unit_decls udecls units in
```

add user defined unit and conversion rule to unit conversion table

Unit System: derived units e.g. m*g/s

Algorithm for Scaling derived units:

1. Decompose units by regular expression
2. Loop and reduce non-basic units like km, cm, ms to basic units like m, s for both sides of assignment
3. Get a map counter of base units, e.g. m*kg /s*s = {“m”: 1, “kg”: 1}, {“s”: 2}
4. Compare counter map of both sides to check if assignment is valid
5. Get conversion rate from two sides

```
203     let derived_get_scale lunit runit =
204       let (sn, sd) = decompose_unit lunit base_units units and (sn', sd') = decompose_unit runit base_units units
205       in
206       let (snb, cn) = reduce_to_base sn and (snb',cn') = reduce_to_base sn' and (sdb,cd) = reduce_to_base sd and (sdb',cd') = reduce_to_base
207       if (StringMap.equal (fun a b -> a = b) snb snb'
208         && StringMap.equal (fun a b -> a = b) sdb sdb')
209         then (cn /. cd )/. (cn' /. cd')
210       else raise( Failure("No conversion rules between unit " ^ lunit ^ " and " ^ runit))
211     in
```

Local variable declare and assign

Local variables inside function declarations:

- Allow Dassign anywhere in the function
- Declare and assign the variable at the same time
- Add to flexibility of the language

```
1  int func main() {
2      int x        = 3;
3      float y      = 0.5;
4      string s     = "foo";
5      bool pred    = true;
6      int[] arr    = [2, 5, 9];
7      float[] farr = [2.5, 3.0, -2.1];
8
9      print(x);
10     printf(y);
11     printl(s);
12     printb(pred);
13     return 0;
14 }
```

```
1  /*locally declare and assign variables*/
2
3  bool func larger(int a, int b) {
4      string ss = "inside compare func";
5      printl(ss);
6      return a > b;
7  }
8
9  int func main() {
10     float '{A} I = 1.2;
11     printl("Second line in main");
12     bool islarger = larger(1, 3);
13     printb(islarger);
14     return 0;
15 }
```

Implementation:

Treat declare-and-assign as a stmt;

Modify expr and stmt by adding a table so that the state can be memorized

Array Manipulation

- Support array initialization, element access, and element modification. Array syntax is similar to C array.
- Support for, while, as well as if-else stmts to better manipulate the array.
- Float array also support unit. Every element in the same array has the same unit. At array initialization, every expr on the right side is assumed no-unit.

```
1  /* example of recording experiment result */
2  int func main() {
3      int i = 0;
4      float[] '{m} dx = [2.3, 4.5, 3.4, 0.7];
5      float[] '{s} dt = [0.5, 0.2, 1.7, 0.5];
6      float[] '{m/s} res = [0.0, 0.0, 0.0, 0.0];
7      for (; i < 4; i = i + 1) {
8          res[i] = dx[i] / dt[i];
9          printf(res[i]);
10     }
11     return 0;
12 }
```

An example of array manipulation w units in a simple physics experiment

4.th

- Overview
- Compiler Architecture
- Features
- Demo
- Teamwork

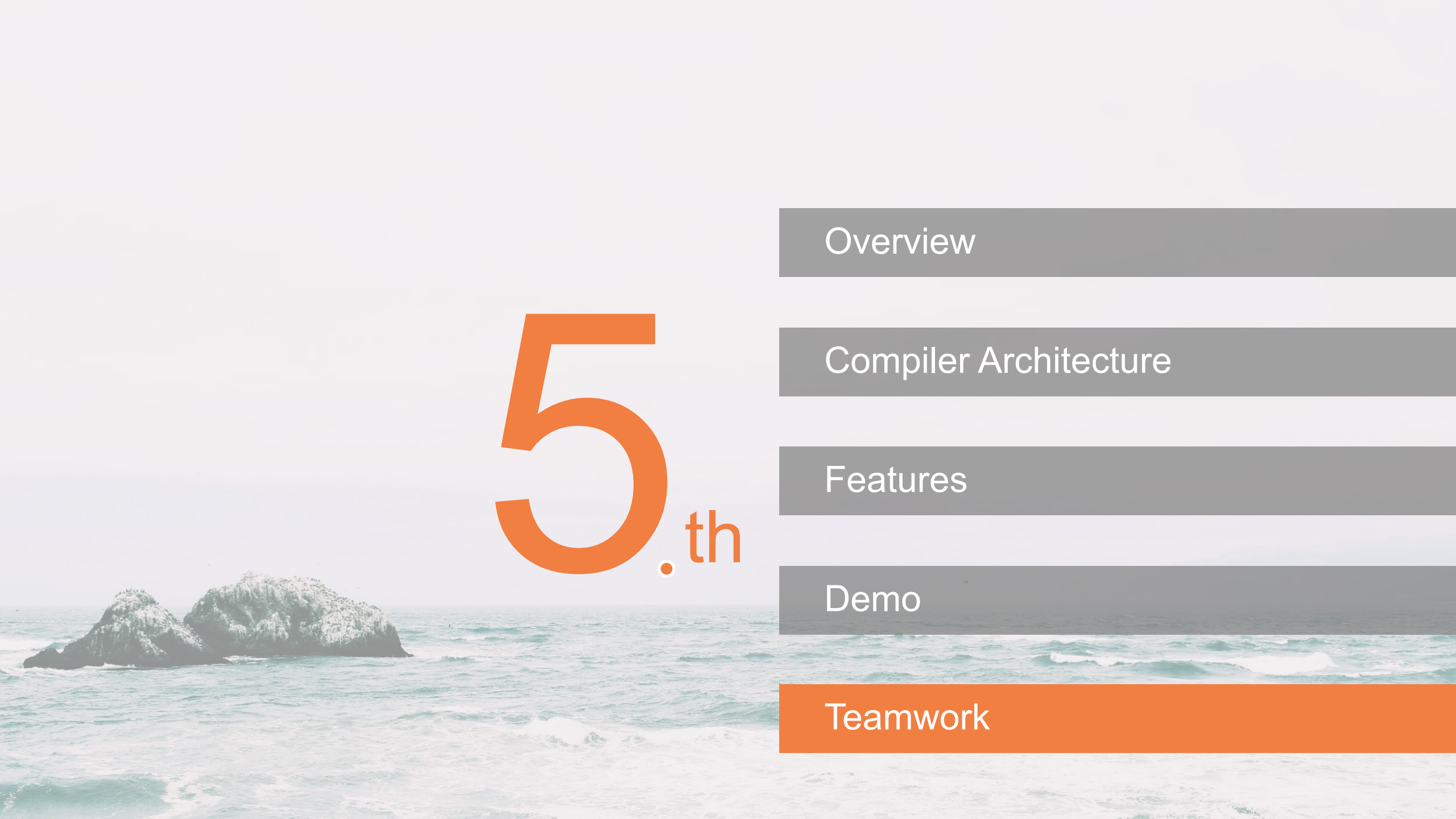
Demo: a comprehensive example

```
/* example of recording experiment result */
|'{mm} = 1000.0 '{m}|;

void func foo(float '{m/s} base) {
    int i = 0;
    float[] '{m} dx = [2.3, 4.5, 3.4, 0.7];
    float[] '{s} dt = [0.5, 0.2, 1.7, 0.5];
    float[] '{mm/s} res = [0.0, 0.0, 0.0, 0.0];
    for (; i < 4; i = i + 1) {
        res[i] = dx[i] / dt[i] + base ;
        /* 2.3m / 0.5s = 4.6 m/s + 0.0122 m/s = 4.6122 m/s = 4612.2 mm/s */
        printf(res[i]);
    }
}

int func main() {
    float '{cm/s} base = 1.22;
    foo(base);                  /* 1.22 cm/s = 0.0122 m/s */
    return 0;
}
```

```
1    4612.2
2+   22512.2
3    2012.2
4    1412.2
5
```


5.th
Overview
Compiler Architecture
Features
Demo
Teamwork

Teamwork

Zhengyuan Dong: Project Manager / Tester

Yucen Sun: Language Guru / Tester

Eleven Li: Compiler Architect / Tester

Future work

- Support for more complicated units (user defined mapping to derived units)
- Equation-like functions.

THANKS

Eleven Li, Yucen Sun, Zhengyuan Dong