Team Project: The Construction of a SASL-Compiler

14.04.2025

Tim Fischer · Nico Faden
{tim.fischer,nico.faden}@uni-tuebingen.de

What is SASL?

SASL (St. Andrews Static Language) is a *pure functional* programming language.

Its features include among others

- arithmetics,
- list processing, and
- recursion.

```
def empty? xs = xs = nil
def map f xs =
   if empty? xs
   then nil
   else (f x : map f xs')
         where x   = hd xs;
               xs' = tl xs
.
map inc [1,2,3,4]
where inc x = x + 1
```

How does it work?

Lexer
Tokenizes
code
into
token stream.

$\rightarrow$

Parser
Parses
token stream
into
parse tree.

$\rightarrow$

Compiler
Compiles
parse tree
into
combinator graph.

$\rightarrow$

Virtual Machine
Evaluates
combinator graph
to
result.

```
31 + x
where x := 11
```

$\rightsquigarrow$ 42

How does it work? — Lexing

Lexer
Tokenizes
code
into
token stream.

$\rightarrow$

Parser
Parses
token stream
into
parse tree.

$\rightarrow$

Compiler
Compiles
parse tree
into
combinator graph.

$\rightarrow$

Virtual Machine
Evaluates
combinator graph
to
result.

```
31 + x
where x := 11
```

$\rightsquigarrow$

`31` > `+` > `x` > `where` > `x` > `:=` > `11` > 🛑

How does it work? — Parsing

Lexer
Tokenizes
code
into
token stream.

→

Parser
Parses
token stream
into
parse tree.

→

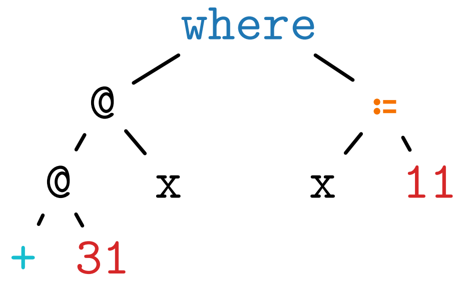
Compiler
Compiles
parse tree
into
combinator graph.

→

Virtual Machine
Evaluates
combinator graph
to
result.

```
31 > + > x > where > x > := > 11 > ⬣
```

$\rightsquigarrow$



How does it work? — Compiling

Lexer
Tokenizes *code* into *token stream.*

→

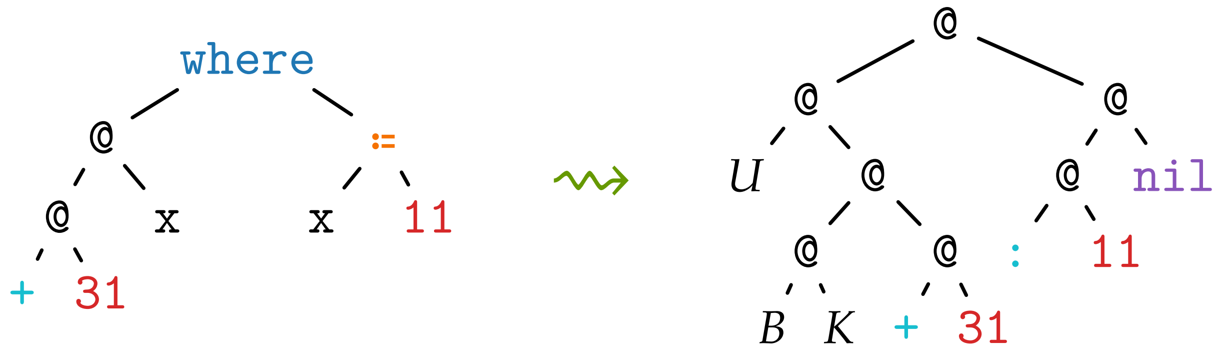
Parser
Parses *token stream* into *parse tree.*

→

Compiler
Compiles *parse tree* into *combinator graph.*

→

Virtual Machine
Evaluates *combinator graph* to *result.*



How does it work? — Evaluating

Lexer
Tokenizes *code* into *token stream*.

→

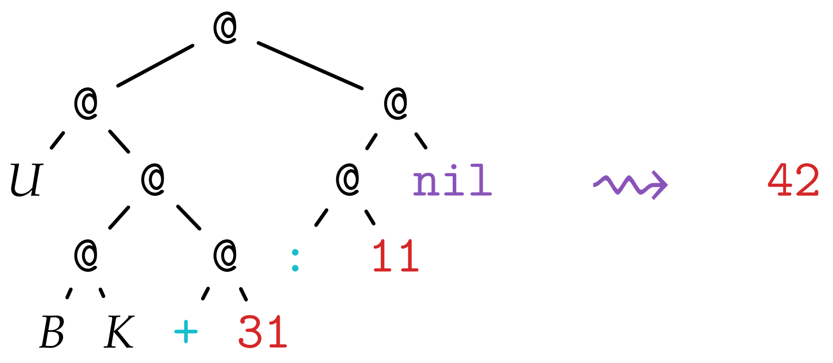
Parser
Parses *token stream* into *parse tree*.

→

Compiler
Compiles *parse tree* into *combinator graph*.

→

Virtual Machine
Evaluates *combinator graph* to *result*.



What is the goal?

You will…

- …build your own *implementation of SASL*…
- …in *teams of two* and…
- …in a *programming language of your choice*.

☝ Some languages make it harder!
Languages without proper *pointers*/*references* require extra steps to represent graphs properly.

👬 It's a *team* project for a reason!
You are to work together! If you feel your partner is lagging behind/running ahead feel free to contact us. And be warned: In extreme cases, either behavior can lead to *failing the course*.

What is your job?

- Apply common programming patterns.
  - We will discuss applicable patterns when necessary.
- Work collaboratively using `git`.
  - You will be allotted GitHub repositories for this purpose.
- Implement and automate unit test.
  - We recommend simply using GitHub Actions for this.

🛠 Version control is your friend.
We assume you *already have* the knowledge and skills necessary for working with `git`.

🤖 Automation is *not* optional!
For most of the commonly used programming languages, you can find project templates that contain a CI/CD configuration for unit tests. *We highly recommend using such a project template!*

What is the plan?

- The project consists of *8 phases*.
  - Each phase focuses on one building block of the project.
- Each phase is *1–2 weeks long*.
  - Lengths vary depending on the complexity, term breaks, *etc*.
- Each phase starts with a *kick-off meeting*.
  - Each on the *first Monday of a phase* at *12:15-13:45* in *B305.1, Sand 13*.
  - We will discuss the contents of the upcoming phase.
- Kick-off meetings are *mandatory*.
  - Not appearing without consulting us is grounds for *failing the course*.

🤔 Hitting a roadblock?
If you ever run out of ideas, have trouble understanding the handout, or just need some a small push to get the ball rolling again, please *contact us!* We are happy to help.

📑 It's all documented.
The kick-off meetings are not your only source of information! We have also provided you with a handout in the initial mail (and in your allotted repository) that details the entire project.

Roadmap

April

Mo	Tu	We	Th	Fr	Sa	Su
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	1	2	3	4

May

Mo	Tu	We	Th	Fr	Sa	Su
28	29	30	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1

June

Mo	Tu	We	Th	Fr	Sa	Su
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	1	2	3	4	5	6

Phase 1	Setup & Exercises	Phase 5	Compiler
Phase 2	Lexer	Phase 6	Virtual Machine
Phase 3	Parse Tree & Visualizer	Phase 7	Optimizer
Phase 4	Parser	Phase 8	Extensions?

1. Find your team partner.
2. Choose your programming language.
3. Get access to your repository.
4. *Get started!*