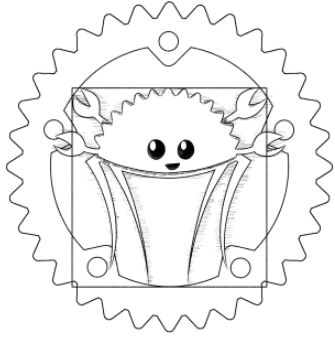
Low-level optimization of algebraic and similar structures

Michalina Kotwica

2020.07.27



What's the plan?

1. type math
2. unit & never
3. structs
4. enums
5. code statistics
6. futures

From "The Book"

5. Using Structs to Structure Related Data

```
struct Point {
  x: i64,
  y: i64,
  color: Color
}
struct Inch(f64)
```

6. Enums and Pattern Matching

```
enum Event {
  Quit,
  KeyPress(char),
  Click {
    x: i64,
    y: i64
  }
}
```

Math underneath

$$\text{Point} = \{\, \text{point}_{x,y,\text{color}} :\ x \in \text{I64} \wedge y \in \text{I64} \wedge \text{color} \in \text{Color} \} \tag{1}$$

$$\text{Point} = \text{I64} \times \text{I64} \times \text{Color} \tag{2}$$

$$\text{Event} = \text{Quit} \cup \text{KeyPress} \cup \text{Click} \tag{3}$$

$$\text{Event} = \{\text{quit}\} \cup \{\, c \in \text{Char} :\ \text{KeyPress}_c \,\} \cup \{\, x, y \in \text{I64} :\ \text{Click}_{x,y} \,\} \tag{4}$$

$$\text{Event} = \{\text{quit}\} \oplus \text{Char} \oplus (\text{I64} \times \text{I64}) \tag{5}$$

$$\text{Event} = 1 + \text{Char} + (\text{I64} \times \text{I64}) \tag{6}$$

Unit

1 {unit} `struct Unit()` ()

```
fn main() -> () {
  (...)
  return;
}
```

`Result<T, ()>` ⟶ `ResultUnitErr { Ok(T), Err }`

Never

0 {} `enum Infallible()` !

```
fn main_loop() -> ! {
  loop {
    (...)
    if error_encountered {
      panic!(...)
    }
  }
}
```

`Result<T, !>` ⟶ `ResultNeverErr { Ok(T) }`

```
let Ok(s) = String::from_str(...);
```

Field reordering

```
struct Foo(u16, u16, u8,  u8)   // 1, 2, 3, 4
struct Bar(u16, u8,  u16, u8)   // 1, 3, 2, 4
struct Bzz(u16, u8, (u16, u8))  // 1, 3,(2, 4)
```

```
C:
  Foo:  11 11  22 22  33 44
  Bar:  11 11  33 xx  22 22  44 xx
  Bzz:  11 11  33 xx (22 22  44 xx)

Rust:
  Foo:  11 11  22 22  33 44
  Bar:  11 11  22 22  33 44
  Bzz:  11 11 (22 22  44 xx) 33 xx
```

Coproduct bool, non-null, non-zero

```
            Option<bool>
                                C        Rust
Some(false)              ...01  ...00     00
Some(true)               ...01  ...01     01
None                     ...00  ...00     02

            Option<Box<T>>, Option<&T>, ...
                                C        Rust
Some(box some_T)         ...01  ...ab     ab
None                     ...00  ...00     00

            Option<NonZeroU8>
                                C        Rust
Some(nonzero!(11u8)) ...01  ...11     ab
None                     ...00  ...00     00
```

Coproduct flattening

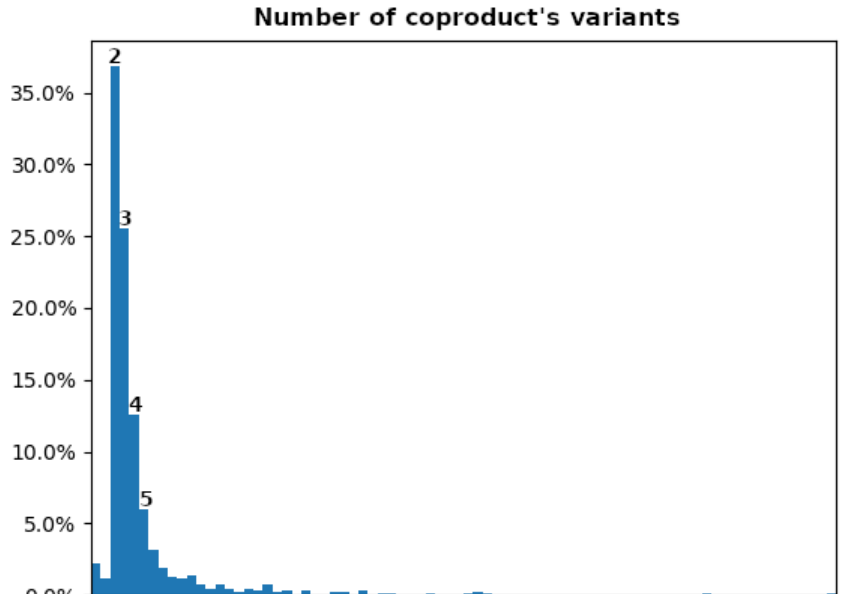
	C	Rust
`Some(u8)`	`...01 ...77`	`01 77`
`None`	`...00 ...00`	`00 00`
`Ok(Some(u8))`	`...01 ...01 ...77`	`01 77`
`Ok(None)`	`...01 ...00 ...00`	`00 00`
`Err(())`	`...00 ...00 ...00`	`02 00`
`Ok(Some(u8))`	`...01 ...01 ...77`	`01 01 77`
`Ok(None)`	`...01 ...00 ...00`	`01 00 00`
`Err(u8)`	`...00 ...44 ...00`	`00 44 00`

Coproduct code scanning

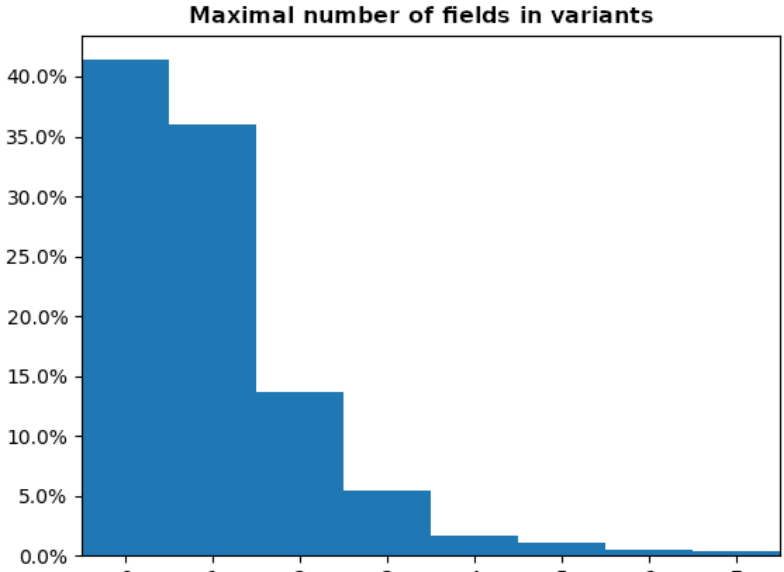
coproduct declarations

- 982 rustc
- 39 serde
- 51 tokio
- 28 syn
- 15 google-apis-rs
- 15 regex
- 5 Octopass
- 2 log

Coproducts stats: variants



Coproducts stats: fields in variant



Coproducts stats: non-empty variants

	0	1	2	3	4	5
0	**41,2%**	**16,4%**	2,4%	1,9%	0,2%	0,0%
1	–	**23,7%**	7,8%	1,2%	0,5%	0,2%
2	–	–	2,7%	0,7%	0,0%	0,2%
3	–	–	–	0,5%	0,0%	0,0%
4	–	–	–	–	0,0%	0,0%
5	–	–	–	–	–	0,0%

```
enum Destination {
  SyncDest,
  AsyncDest,
}
```

```
enum Poll<T> {
  Ready(T),
  Pending,
}
```

```
enum Either<A, B> {
  Left(A),
  Right(B),
}
```

Futures: coproducts on steroids

```
async {
  while let Some(signal) = signals.next().await {
    signal match {
      Msg((id, data)) => target[id].send(data).await,
      Shutdown(err) => return Err(err)
    }
  }
  Ok(())
}
```

```
enum BlockFuture {
  Enter       { signals: S },
  SignalsNext { signals: S },
  TargetSend  { signals: S, id: usize, data: Data },
  Error(Error),
  Return
} // E -> SN;  SN -> TS | E | R;  TS -> SN | E
```

Futures: coproducts on steroids

```
SomeFuture {
  vec: Vec<i32>,          // 24
  a:   usize,             //  8
  b:   usize,             //  8
  it:  Iter<Vec<i32>>,    // 16
}
```

	vec	a	b	it
0	x			
1	x	x	x	
2	x			x
3	x			

```
vvvvvvvv vvvvvvvv vvvvvvvv AAAAAAAA BBBBBBBB

vvvvvvvv vvvvvvvv vvvvvvvv IIIIIIII IIIIIIII
```

Questions

Questions?

Remarks?

Comments?