REWRITING TYPES

MICHAL J. GAJDA

MIGAMAKE PTE LTD

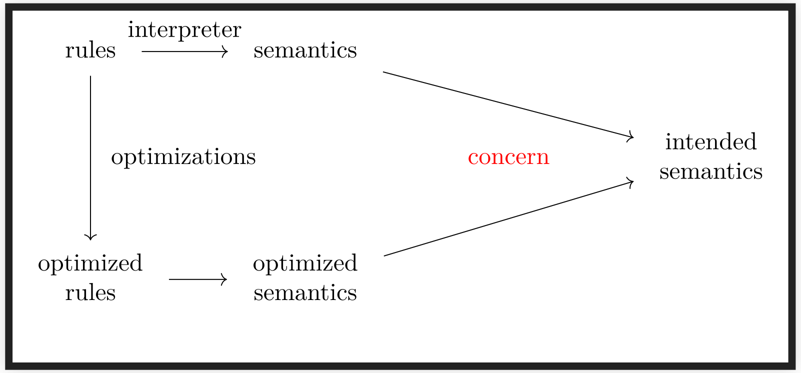
Why rewriting for higher order types?

July 2019

APP ENGINE

- Term rewriting system
- Terminating and co-inductive types only
- Confluence checks
  - detect critical pairs by unification of heads
  - quickcheck
  - explicit rule sequence
- higher order (implicit curry)

OPTIMIZABLE SEMANTICS



Concern is equalizer of semantics.

DIVIDE-AND-CONQUER

…fears and concerns

Separate:

- type safety
- data security
- protocol safety
- correct answers

VALIDATION BY TYPECHECKING

Examples:

- Hindley-Milner - type safety
- session types - protocol adherence
- array shape types - linear algebra correctness
- physical unit types - physical unit matching
- computational complexity properties - predictable runtime

SIMPLE TYPE SYSTEM

Types:

```
Zero :: -> Int
Succ :: Int -> Int

Int /= String
```

Program:

```
mul (Succ y) x = add x (mul y x)
mul  Zero    x = Zero
add  Zero    x = x
add (Succ y) x = add y (Succ x)
```

Types:

```
Zero :: -> Int
Succ :: Int -> Int
```

Program:

```
mul (Succ y) x = add x (mul y x)
mul  Zero    x = Zero
```

Acceptable:

```
mul :: Int Int -> Int
add :: Int Int -> Int
```

Types:

```
Zero :: -> Int
Succ :: Int -> Int
```

Program:

```
mul (Succ y) x = add x (mul y x)
mul  Zero    x = Zero
```

Unacceptable

```
mul :: Int String -> Int
```

GOALS

- Type system as library TRS
- Optimizations as library TRS
- Validation in common framework