"Good" vs. "Bad" Expressions

```
; interp-expr FAE ... -> FAE-Value
```

"Good" vs. "Bad" Expressions

```
; interp-expr FAE ... -> FAE-Value
```

- Does `interp-expr` produce a value for all expressions?

"Good" vs. "Bad" Expressions

```
; interp-expr FAE ... -> FAE-Value
```

- Does **`interp-expr`** produce a value for all expressions?
- Of course not!

"Good" vs. "Bad" Expressions

```
; interp-expr FAE ... -> FAE-Value
```

- Does `interp-expr` produce a value for all expressions?
- Of course not!
- `(interp-expr (parse '{5 5}))` etc ...

"Good" vs. "Bad" Expressions

```
; interp-expr FAE ... -> FAE-Value
```

- Does `interp-expr` produce a value for all expressions?
- Of course not!
- `(interp-expr (parse '{5 5}))` etc ...
- But do we know enough about expressions to tell before actually calling `interp-expr`?

Quiz

- Question #1: What is the value of the following expression?

```
{+ 1 2}
```

Quiz

- Question #1: What is the value of the following expression?

```
{+ 1 2}
```

- Wrong answer: **0**

Quiz

- Question #1: What is the value of the following expression?

```
{+ 1 2}
```

- Wrong answer: **0**
- Wrong answer: **42**

Quiz

- Question #1: What is the value of the following expression?

```
{+ 1 2}
```

- Wrong answer: **0**
- Wrong answer: **42**

- Answer: **3**

Quiz

- Question #2: What is the value of the following expression?

```
{+ fun 17 8}
```

Quiz

- Question #2: What is the value of the following expression?

```
{+ fun 17 8}
```

- Wrong answer: **error**

Quiz

- Question #2: What is the value of the following expression?

```
{+ fun 17 8}
```

- Wrong answer: **error**

- Answer: Trick question! `{+ fun 17 8}` is not an expression

Language Grammar for Quiz

```
<MFAE> ::= <num>
        |  true
        |  false
        |  {+ <MFAE> <MFAE>}
        |  {- <MFAE> <MFAE>}
        |  {= <MFAE> <MFAE>}
        |  <id>
        |  {fun {<id>*} <MFAE>}
        |  {<MFAE> <MFAE>*}
        |  {if <MFAE> <MFAE> <MFAE>}
```

Quiz

- Question #3: Is the following an expression?

```
{{fun {x y} 1} 7}
```

Quiz

- Question #3: Is the following an expression?

```
{{fun {x y} 1} 7}
```

- Wrong answer: **No**

Quiz

- Question #3: Is the following an expression?

```
{{fun {x y} 1} 7}
```

- Wrong answer: **No**

- Answer: **Yes** (according to our grammar)

Quiz

- Question #4: What is the value of the following expression?

```
{{fun {x y} 1} 7}
```

Quiz

- Question #4: What is the value of the following expression?

```
{{fun {x y} 1} 7}
```

- Answer: `{fun {y} 1}` (according to some interpreters)

Quiz

- Question #4: What is the value of the following expression?

  `{{fun {x y} 1} 7}`

- Answer: `{fun {y} 1}` (according to some interpreters)

- But no *real* language would accept `{{fun {x y} 1} 7}`

Quiz

- Question #4: What is the value of the following expression?

  `{{fun {x y} 1} 7}`

- Answer: `{fun {y} 1}` (according to some interpreters)

- But no *real* language would accept `{{fun {x y} 1} 7}`

- Let's agree to call `{{fun {x y} 1} 7}` an ***ill-formed expression*** because `{fun {x y} 1}` should be used only with two arguments

- Let's agree to never evaluate ill-formed expressions

Quiz

- Question #5: What is the value of the following expression?

```
{{fun {x y} 1} 7}
```

Quiz

- Question #5: What is the value of the following expression?

```
{{fun {x y} 1} 7}
```

- Answer: **None** - the expression is ill-formed

Quiz

- Question #6: Is the following a well-formed expression?

```
{+ {fun {} 1} 8}
```

Quiz

- Question #6: Is the following a well-formed expression?

```
{+ {fun {} 1} 8}
```

- Answer: **Yes**

Quiz

- Question #7: What is the value of the following expression?

```
{+ {fun {} 1} 8}
```

Quiz

- Question #7: What is the value of the following expression?

```
{+ {fun {} 1} 8}
```

- Answer: **None** - it produces an error:

  numeric operation expected number

Quiz

- Question #7: What is the value of the following expression?

```
{+ {fun {} 1} 8}
```

- Answer: **None** - it produces an error:

  numeric operation expected number

- Let's agree that a `fun` expression cannot be inside a `+` form

Quiz

- Question #8: Is the following a well-formed expression?

```
{+ {fun {} 1} 8}
```

Quiz

- Question #8: Is the following a well-formed expression?

```
{+ {fun {} 1} 8}
```

- Answer: **No**

Quiz

- Question #9: Is the following a well-formed expression?

```
{+ {{fun {x} x} 7} 5}
```

Quiz

- Question #9: Is the following a well-formed expression?

```
{+ {{fun {x} x} 7} 5}
```

- Answer: Depends on what we meant by *inside* in our most recent agreement
  - *Anywhere inside* - **No**
  - *Immediately inside* - **Yes**

Quiz

- Question #9: Is the following a well-formed expression?

```
{+ {{fun {x} x} 7} 5}
```

- Answer: Depends on what we meant by *inside* in our most recent agreement
  - *Anywhere inside* - **No**
  - *Immediately inside* - **Yes**

- Since our intrepreter produces **12**, and since that result makes sense, let's agree on *immediately inside*

Quiz

- Question #10: Is the following a well-formed expression?

```
{+ {{fun {x} x} {fun {y} y}} 5}
```

Quiz

- Question #10: Is the following a well-formed expression?

```
{+ {{fun {x} x} {fun {y} y}} 5}
```

- Answer: **Yes**, but we don't want it to be!

Quiz

- Question #11: Is it possible to define ***well-formed*** (as a decidable property) so that we reject all expressions that produce errors?

Quiz

- Question #11: Is it possible to define ***well-formed*** (as a decidable property) so that we reject all expressions that produce errors?

- Answer: **Yes**: reject *all* expressions!

Quiz

- Question #12: Is it possible to define ***well-formed*** (as a decidable property) so that we reject *only* expressions that produce errors?

Quiz

- Question #12: Is it possible to define ***well-formed*** (as a decidable property) so that we reject *only* expressions that produce errors?

- Answer: **No**

Quiz

- Question #12: Is it possible to define ***well-formed*** (as a decidable property) so that we reject *only* expressions that produce errors?

- Answer: **No**

```
{+ 1 {if ... 1 {fun {x} x}}}
```

- If we always knew whether `...` produces true or false, we could solve the halting problem

Types

- Solution to our dilemma
  - In the process of rejecting expressions that are certainly bad, also reject some expressions that are good

```
{+ 1
   {if {prime? 131101}
       1
       {fun {x} x}}}
```

Types

- Overall strategy:
    - Assign a ***type*** to each expression *without evaluating*
    - Compute the type of a complex expression based on the types of its subexpressions

Types

1 : ***num***

true : ***bool***

Types

```
1 : num

true : bool

{+ 1 2}
```

Types

```
1 : num

true : bool

{+ 1 2}
num
```

Types

`1 : num`

`true : bool`

`{+ 1 2}`

num *num*

Types

`1 : num`

`true : bool`

`{+ 1 2}`

`num` `num`

`num`

Types

`1 : num`

`true : bool`

`{+ 1 2}`

num *num*

num

`{+ 1 false}`

Types

```
1 : num

true : bool

{+ 1 2}
num      num
    num

{+ 1 false}
num
```

Types

```
1 : num

true : bool

{+ 1 2}
num        num
     num

{+ 1 false}
num       bool
```

Types

`1 : num`

`true : bool`

`{+ 1 2}`

num *num*

num

`{+ 1 false}`

num *bool*

no type