Introduction to Contracts

Paulo Matos <pmatos@linki.tools>

FLAT CONTRACTS

- Predicates are flat contracts: **`vector?`**, **`list?`**, **`symbol?`**...

FLAT CONTRACTS

- Predicates are flat contracts: **`vector?`**, **`list?`**, **`symbol?`**...

```
(define/contract an-integer
  integer?
  2)
(define/contract a-list
  list?
  '(1 2 3))
```

CONTRACT COMBINATORS - 1

- **`list?`** checks for lists but how to check for lists of integers?

CONTRACT COMBINATORS - 1

- **`list?`** checks for lists but how to check for lists of integers?
  - **`(listof integer?)`**

CONTRACT COMBINATORS - 1

- **`list?`** checks for lists but how to check for lists of integers?
  - **`(listof integer?)`**
- vectors of symbols:

CONTRACT COMBINATORS - 1

- **`list?`** checks for lists but how to check for lists of integers?
  - **`(listof integer?)`**
- vectors of symbols:
  - **`(vectorof symbol?)`**

CONTRACT COMBINATORS - 1

- **`list?`** checks for lists but how to check for lists of integers?
  - **`(listof integer?)`**
- vectors of symbols:
  - **`(vectorof symbol?)`**
- pair of symbol and boolean:

CONTRACT COMBINATORS - 1

- **`list?`** checks for lists but how to check for lists of integers?
  - **`(listof integer?)`**
- vectors of symbols:
  - **`(vectorof symbol?)`**
- pair of symbol and boolean:
  - **`(cons/c symbol? boolean?)`**

CONTRACT COMBINATORS - 1

- **`list?`** checks for lists but how to check for lists of integers?
  - **`(listof integer?)`**
- vectors of symbols:
  - **`(vectorof symbol?)`**
- pair of symbol and boolean:
  - **`(cons/c symbol? boolean?)`**
- set of pairs of integer

CONTRACT COMBINATORS - 1

- **`list?`** checks for lists but how to check for lists of integers?
  - **`(listof integer?)`**
- vectors of symbols:
  - **`(vectorof symbol?)`**
- pair of symbol and boolean:
  - **`(cons/c symbol? boolean?)`**
- set of pairs of integer
  - **`(set/c (cons/c integer? integer?))`**

CONTRACT COMBINATORS - 2

- Function contracts:
  - ***ctc-domain1*** ... ***ctc-domainN*** . -> . ***range***

CONTRACT COMBINATORS - 2

- Function contracts:
  - ***ctc-domain1*** … ***ctc-domainN*** . -> . ***range***
  - **(integer? #:x 0 . -> . integer?)**

CONTRACT COMBINATORS - 2

- Function contracts:
  - ***ctc-domain1*** … ***ctc-domainN*** . -> . ***range***
  - **(integer? #:x 0 . -> . integer?)**
  - **((listof integer?) . -> . (values symbol? boolean?))**

CONTRACT COMBINATORS - 2

- Function contracts:
  - ***ctc-domain1*** … ***ctc-domainN*** . -> . ***range***
  - **(integer? #:x 0 . -> . integer?)**
  - **((listof integer?) . -> . (values symbol? boolean?))**
  - **symbol? ... . -> . void?**

CONTRACT COMBINATORS - 2

- Function contracts:
  - ***ctc-domain1*** … ***ctc-domainN*** . -> . ***range***
  - **(integer? #:x 0 . -> . integer?)**
  - **((listof integer?) . -> . (values symbol? boolean?))**
  - **symbol? ... . -> . void?**
  - **real? ... symbol? . -> . symbol?**

CONTRACT COMBINATORS - 3

- Now with optional arguments and rest arguments
  - ```
    (->* (mandatory) (optional) #:rest (rest)
         range)
    ```
    ```

CONTRACT COMBINATORS - 3

- Now with optional arguments and rest arguments
  - ```
    (->* (mandatory) (optional) #:rest (rest)
         range)
    ```
  - ```
    (->* (integer?) (#:id symbol?) #:rest (listof real?)
         (vector/c integer? boolean?))
    ```

CONTRACT COMBINATORS - 3

- Now with optional arguments and rest arguments
  - ```
    (->* (mandatory) (optional) #:rest (rest)
         range)
    ```
  - ```
    (->* (integer?) (#:id symbol?) #:rest (listof real?)
         (vector/c integer? boolean?))
    ```
  - ```
    (->* (boolean?) (symbol? symbol?)
         void?)
    ```
    ```

CONTRACT COMBINATORS - 4

- Now with relationships
  - `->i` same as `->*` but each member of the domain or range is named to be referenced elsewhere

CONTRACT COMBINATORS - 4

- Now with relationships
  - **`->i`** same as **`->*`** but each member of the domain or range is named to be referenced elsewhere
  - ```
    (->i ([a (listof integer?)]
          [b (a) (and/c (listof boolean?)
                        (= (length a) (length b)))])
         (boolean?)
         [r (a b)
            (and/c (listof symbol?)
                   (= (length r)
                      (abs (- (length a) (length b)))))])
    ```
    ```

CONTRACT COMBINATORS - 4

- Now with relationships
  - **`->i`** same as **`->*`** but each member of the domain or range is named to be referenced elsewhere
  - ```
    (->i ([a (listof integer?)]
          [b (a) (and/c (listof boolean?)
                        (= (length a) (length b)))])
         (boolean?)
         [r (a b)
            (and/c (listof symbol?)
                   (= (length r)
                      (abs (- (length a) (length b)))))])
    ```
  - ```
    (->i ([a integer?]
          [b (a)
             (and/c integer? (>/c b a))])
         integer?)
    ```

CONTRACT COMBINATORS - 5

- Structure instance

```
(struct complex (r i))
```

CONTRACT COMBINATORS - 5

- Structure instance

```
(struct complex (r i))

(struct/c complex zero? zero?)
```

CONTRACT COMBINATORS - 5

- Structure instance

```
(struct complex (r i))

(struct/c complex zero? zero?)
```

- Structural information

```
(struct complex
  ([r real?]
   [i real?]))
```

EXERCISE - 1

code/3-function-ds-contracts/exercises.rkt

BREAKING CONTRACTS - 1

BREAKING CONTRACTS - 1

```
(define/contract (sum-list xs)
  ((listof integer?) . -> . integer?)
  (define total
    (for/fold ([s 0])
              ([x (in-list xs)])
      (+ s x)))
  (if (> total 100)
      'buh
      total))
```

BREAKING CONTRACTS - 2

```
(contract-exercise sum-list)
```

EXERCISE - 2

- Can we create a function with a contract, where the function breaks the contract but for which `contract-exercise` does not find any problem with it?

CONTRACT PROFILING

- Open **`code/5-profiling/5.0-mandel/`**
  - take some time to look at the code
- How much time is taken by contracts?
  - 10%?

CONTRACT PROFILING

- Open **`code/5-profiling/5.0-mandel/`**
  - take some time to look at the code
- How much time is taken by contracts?
  - 10%?
  - 20%?

CONTRACT PROFILING

- Open **`code/5-profiling/5.0-mandel/`**
  - take some time to look at the code
- How much time is taken by contracts?
  - 10%?
  - 20%?
  - 30%?

CONTRACT PROFILING

- Open **`code/5-profiling/5.0-mandel/`**
  - take some time to look at the code
- How much time is taken by contracts?
  - 10%?
  - 20%?
  - 30%?
  - 40%?

CONTRACT PROFILING

- Open **`code/5-profiling/5.0-mandel/`**
  - take some time to look at the code
- How much time is taken by contracts?
  - 10%?
  - 20%?
  - 30%?
  - 40%?
  - 50%?

CONTRACT PROFILING

- Open **`code/5-profiling/5.0-mandel/`**
  - take some time to look at the code
- How much time is taken by contracts?
  - 10%?
  - 20%?
  - 30%?
  - 40%?
  - 50%?
  - 60%?

CONTRACT PROFILING

- Open **`code/5-profiling/5.0-mandel/`**
  - take some time to look at the code
- How much time is taken by contracts?
  - 10%?
  - 20%?
  - 30%?
  - 40%?
  - 50%?
  - 60%?
  - 70%?

CONTRACT PROFILING

- Open **`code/5-profiling/5.0-mandel/`**
  - take some time to look at the code
- How much time is taken by contracts?
  - 10%?
  - 20%?
  - 30%?
  - 40%?
  - 50%?
  - 60%?
  - 70%?
  - 80%?

EXERCISE - 3

- Use `contract-profile-thunk` to profile the contracts

DISABLING CONTRACTS - EXERCISE 4

- Take a look at **`code/5-profiling/5.2-disabling-contracts`**
  - Run with and without contracts by changing the value of the variable **`ENABLE_CONTRACTS`**

THANK YOU!