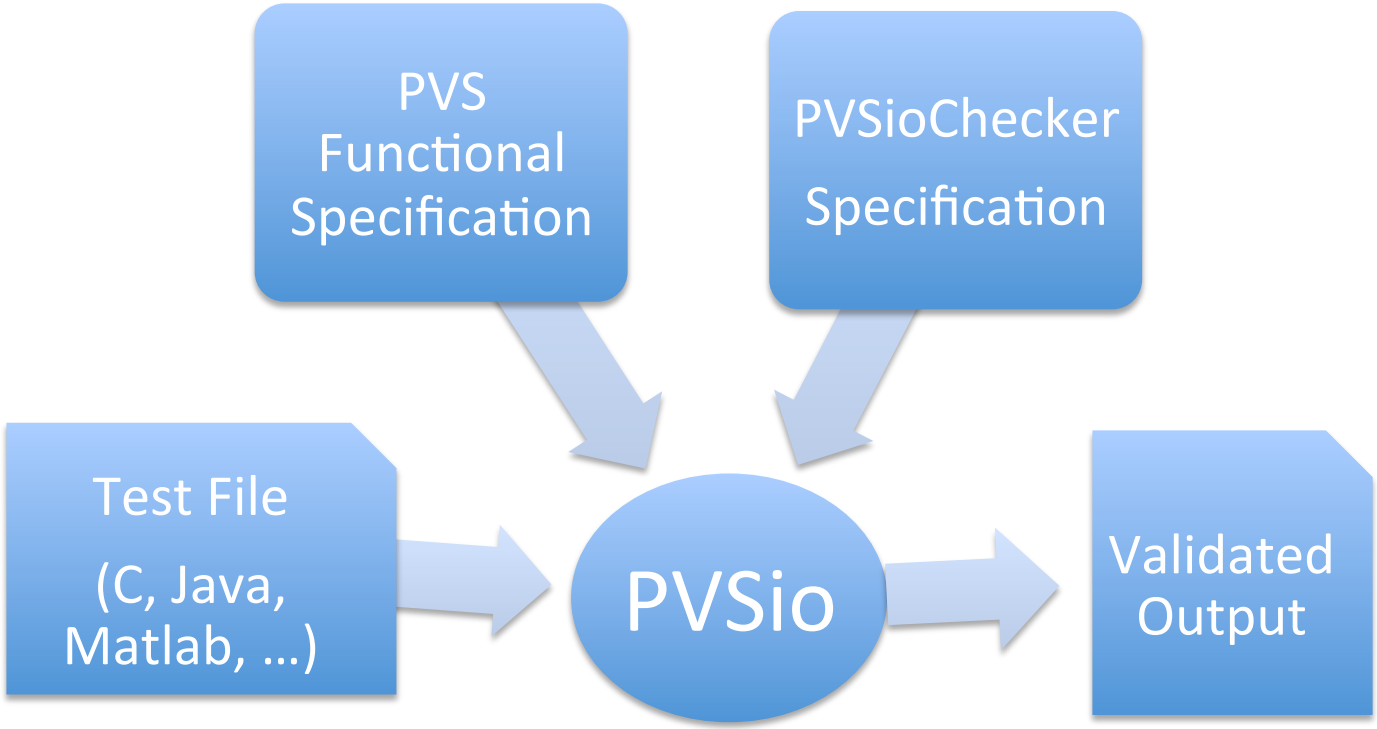
PVSioChecker

Semantic Validation in PVS

High Level View



PVS Functional Specification

```
% Naive primality test

is_prime?(n:nat) : bool =
  n > 1 AND (n = 2 OR
  FORALL (j:subrange(2,n-1)): mod(n,j) /= 0)
```

Test File: `primes.txt`

```
1
TRUE
5
TRUE
9
FALSE
10
FALSE
9
FALSE
3
TRUE
17
FALSE
```

Record 1

Record 2

Record n

Records

```
10      % Input
FALSE   % Computed output by C, Java, etc
```

- A record may have any number of lines, inputs and outputs, but all records in a file should have the same structure, i.e., number of lines, number of input/outputs, etc.
- Inputs are provided one per line in PVS syntax.
- Outputs are provided one per line in PVS syntax.
- Empty lines and PVS comments are ignored.

PVSioChecker Specification

```
prime_check : THEORY
BEGIN
  IMPORTING PVSioChecker@pvsio_checker,
            is_prime

check?(fin:IStream,fout:OStream) : bool =
  LET
    n   = str2pvs[nat](readln_checker(fin)),
    io  = str2pvs[bool](readln_checker(fin)),
    pvs = is_prime?(n) IN
      check_bool(fout,"",pvs,io)

main(file:string,records:nat): void =
  checker(file,check?,records)

END prime_check
```

PVSio from PVS (`M-x pvsio`)

- `<PVSio> main("<filename>",<n>)`
- where
  - `<filename>` is a the name of the file to be checked (or empty string for standard input),
  - `<n>` is the number of records to be checker (or 0 for all records).

```
<PVSio> main("primes.txt",0);
Reading: primes.txt. Writing: primes.out
[51%] Estimated Time of Completion: 0h:0m:0.073s
[61%] Estimated Time of Completion: 0h:0m:0.087s
[72%] Estimated Time of Completion: 0h:0m:0.075s
[83%] Estimated Time of Completion: 0h:0m:0.051s
[93%] Estimated Time of Completion: 0h:0m:0.021s
Real time: 0.344 sec. Run time: 0.350 sec
Lines: 122. Records: 56. Fails: 16
```

PVSio from Command Line

- `$ pvsio @<theory>: \"<file>\" <n>`
- Where
  - `<theory>` is the name of the theory where the function main is defined,
  - `<file>` and `<n>` as before. Double quotes need to be escaped.

```
$ pvsio @prime_check: \"primes.txt\" 0
Reading: primes.txt. Writing: primes.out
[17%] Estimated Time of Completion: 0h:0m:0.150s
[35%] Estimated Time of Completion: 0h:0m:0.133s
[53%] Estimated Time of Completion: 0h:0m:0.086s
[71%] Estimated Time of Completion: 0h:0m:0.050s
[89%] Estimated Time of Completion: 0h:0m:0.018s
Real time: 0.167 sec. Run time: 0.170 sec
Lines: 112. Records: 56. Fails: 16
```

Validated Output

```
Date: Tuesday June 24 2014, 10:10:54 (GMT-5)
Input file: primes.txt
*** ERROR. Line: 2. Record: 1. PVS: FALSE vs. Input: TRUE
*** ERROR. Line: 14. Record: 7. PVS: TRUE vs. Input: FALSE
*** ERROR. Line: 16. Record: 8. PVS: FALSE vs. Input: TRUE
*** ERROR. Line: 28. Record: 14. PVS: TRUE vs. Input: FALSE
*** ERROR. Line: 30. Record: 15. PVS: FALSE vs. Input: TRUE
*** ERROR. Line: 42. Record: 21. PVS: TRUE vs. Input: FALSE
…
*** ERROR. Line: 86. Record: 43. PVS: FALSE vs. Input: TRUE
*** ERROR. Line: 98. Record: 49. PVS: TRUE vs. Input: FALSE
*** ERROR. Line: 100. Record: 50. PVS: FALSE vs. Input: TRUE
*** ERROR. Line: 112. Record: 56. PVS: TRUE vs. Input: FALSE
Real time: 0.167 sec. Run time: 0.170 sec
Lines: 112. Records: 56. Fails: 16
```

Additional Features

- Validating a limited number of records:

```
$ pvsio @prime_check: \"primes.txt\" 10

Reading: primes.txt. Writing: primes-10.out
[100%] Estimated Time of Completion: 0h:0m:0.000s
Real time: 0.079 sec. Run time: 0.080 sec
Lines: 30. Records: 10. Fails: 3
```

Additional Features

- Checking integers and real numbers:
  - `check_int(fout,str,pvs,io): str` is an error message, `pvs` and `io` are integers.
  - `check_real(fout,str,pvs,io): str` is an error message, `pvs` and `io` are real numbers. Numbers are checked with respect to a default precision.
  - `check_real_prec(fout,str,pvs,io,prec): str` is an error message, `pvs` and `io` are real numbers. Numbers are checked against precision `prec`, which is a non-negative real number.

```
<PVSio> check_real(stdout,"",sq(sqrt(2)),2);
==>
TRUE

<PVSio> check_real_prec(stdout,": Numbers are not equal",sq(sqrt(2)),2,0);
*** ERROR: Numbers are not equal. Line: 122. Record: 56. PVS: 1.999 vs. Input: 2
==>
FALSE
```