LWC'14: Rascal

Tijs van der Storm @tvdstorm
Pablo Inostroza Valdera @metalinguist

Rascal in a nutshell

- Functional meta programming language
- Built-in context free grammars

- Primitives for: traversal, matching, generation, querying, and analysis.
- IDE hooks into Eclipse: language workbench

Rascal QL “2.0”

- Uses reference implementation
- Roughly 450 SLOC
- Includes: grammar, AST, outliner, type checker, error marking, name resolution, hyperlinking
- No cycle detection, no QLS

```
form taxOfficeExample {
  "Did you buy a house in 2010?"
    hasBoughtHouse: boolean

  "Did you enter a loan?"
    hasMaintLoan: boolean

  "Did you sell a house in 2010?"
    hasSoldHouse: boolean

  if (hasSoldHouse) {
    "What was the selling price?"
      sellingPrice: money
    "Private debts for the sold house:"
      privateDebt: money
    "Value residue:"
      valueResidue: money
        = sellingPrice - privateDebt
  }
}
```

Scalability

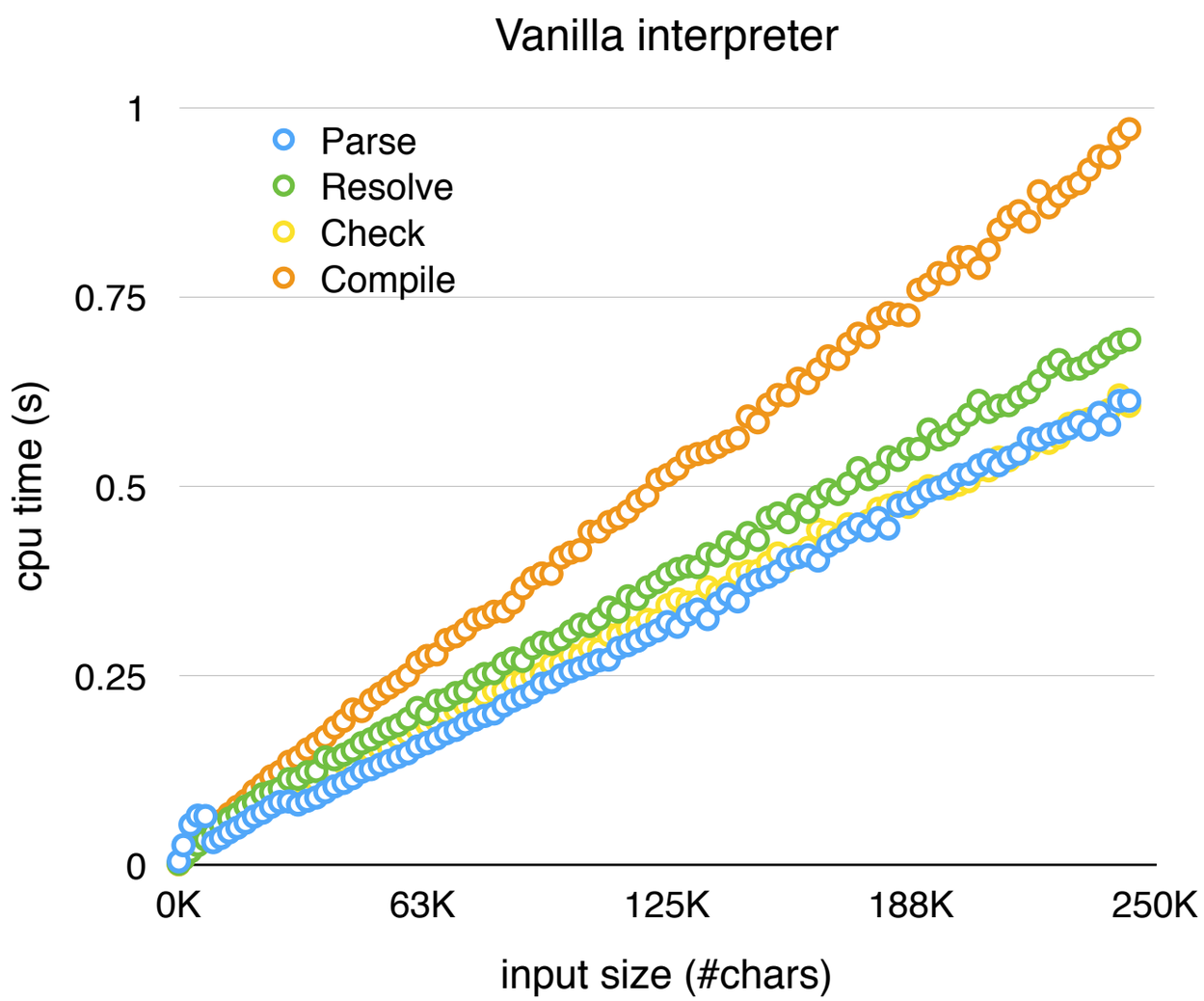
Generating binary search

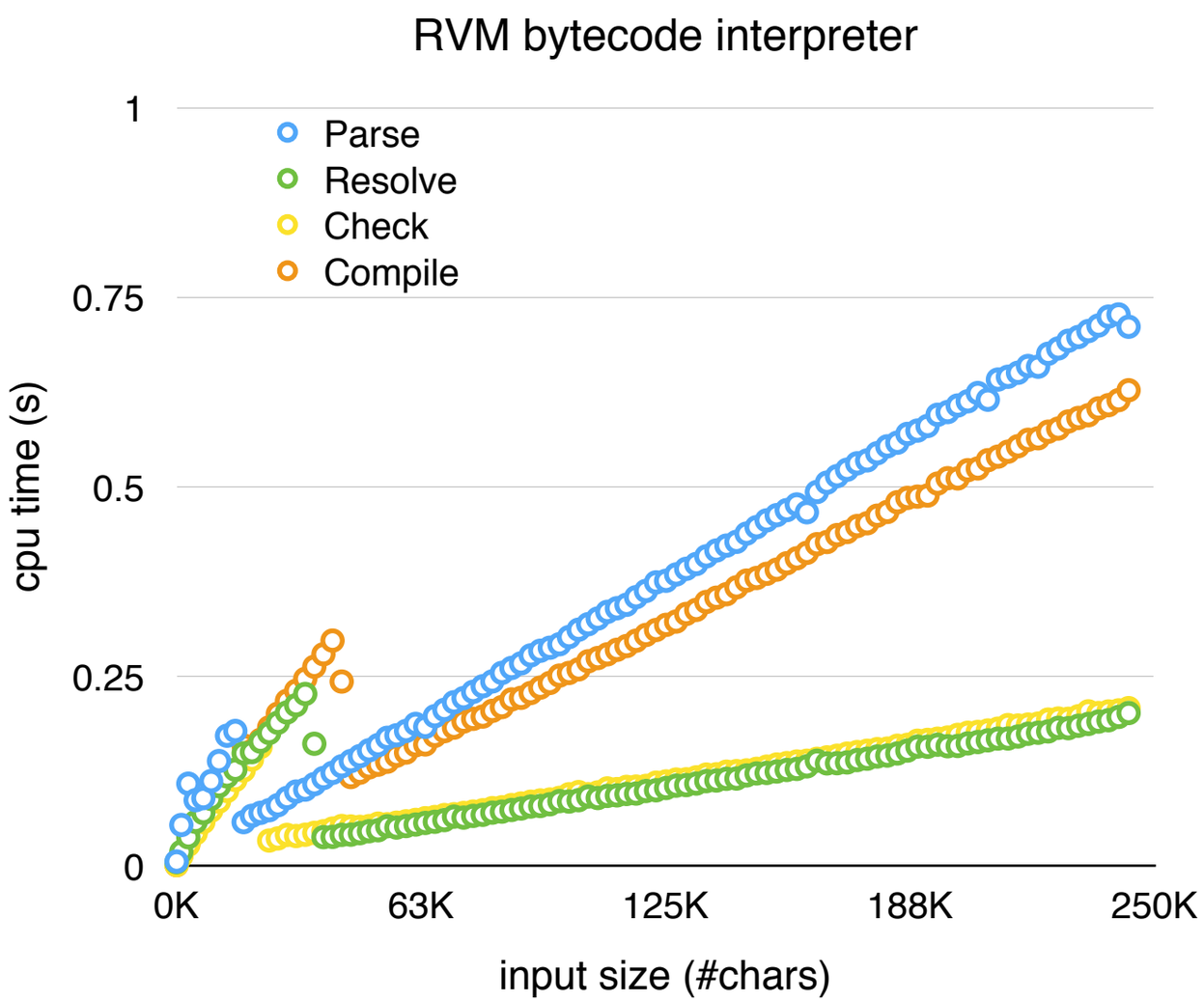
```
str binForm(int min, int max)
  = "form binary {
    '  <binFormRec(min, max)>
    '}";

str binFormRec(int min, int max)
  = "\"The answer is\" answer_<min>_<max>: integer = (<min>)"
  when max - min <= 1;

default str binFormRec(int min, int max)
  = "\"Between <min> and <half>?\" x_<min>_<half>: boolean
    'if (x_<min>_<half>) {
    '  <binFormRec(min, half)>
    '}
    'else {
    '  <binFormRec(half, max)>
    '}"
  when half := min + ((max - min) / 2);
```

```
form binary {
  "Is the number between 1 and 10" x_1_10: boolean
  if (x_1_10) {
    "Is the number between 1 and 5" x_1_5: boolean
    if (x_1_5) {
      "Is the number between 1 and 3" x_1_3: boolean
      if (x_1_3) {
        "Is the number between 1 and 2" x_1_2: boolean
        if (x_1_2) {
          "The answer is" answer_1_2: integer = (1)
        }
        else {
          "The answer is" answer_2_3: integer = (2)
        }
      }
      else {
        "Is the number between 3 and 4" x_3_4: boolean
        if (x_3_4) {
          "The answer is" answer_3_4: integer = (3)
        }
        else {
          "The answer is" answer_4_5: integer = (4)
        }
      }
    }
    else {
      "Is the number between 5 and 7" x_5_7: boolean
      if (x_5_7) {
        "Is the number between 5 and 6" x_5_6: boolean
        if (x_5_6) {
          "The answer is" answer_5_6: integer = (5)
        }
        else {
          "The answer is" answer_6_7: integer = (6)
        }
      }
      else {
        "Is the number between 7 and 8" x_7_8: boolean
        if (x_7_8) {
          "The answer is" answer_7_8: integer = (7)
        }
        else {
          "Is the number between 8 and 9" x_8_9: boolean
          if (x_8_9) {
            "The answer is" answer_8_9: integer = (8)
          }
          else {
            "The answer is" answer_9_10: integer = (9)
          }
        }
      }
    }
  }
  else {
    "Is the number between 10 and 15" x_10_15: boolean
    if (x_10_15) {
      "Is the number between 10 and 12" x_10_12: boolean
      if (x_10_12) {
        "Is the number between 10 and 11" x_10_11: boolean
        if (x_10_11) {
          "The answer is" answer_10_11: integer = (10)
        }
        else {
          "The answer is" answer_11_12: integer = (11)
        }
      }
      else {
        "Is the number between 12 and 13" x_12_13: boolean
        if (x_12_13) {
          "The answer is" answer_12_13: integer = (12)
        }
        else {
          "Is the number between 13 and 14" x_13_14: boolean
          if (x_13_14) {
            "The answer is" answer_13_14: integer = (13)
          }
          else {
            "The answer is" answer_14_15: integer = (14)
          }
        }
      }
    }
    else {
      "Is the number between 15 and 17" x_15_17: boolean
      if (x_15_17) {
        "Is the number between 15 and 16" x_15_16: boolean
        if (x_15_16) {
          "The answer is" answer_15_16: integer = (15)
        }
        else {
          "The answer is" answer_16_17: integer = (16)
        }
      }
      else {
        "Is the number between 17 and 18" x_17_18: boolean
        if (x_17_18) {
          "The answer is" answer_17_18: integer = (17)
        }
        else {
          "Is the number between 18 and 19" x_18_19: boolean
          if (x_18_19) {
            "The answer is" answer_18_19: integer = (18)
          }
          else {
            "The answer is" answer_19_20: integer = (19)
          }
        }
      }
    }
  }
}
```

Benchmark

- Run: parse, resolve, check, compile (PRCC)
- On “binary” where 1 <= n <= 1000 (step 10)
- Measure cpu-time
  - Vanilla interpreter
  - Rascal Virtual Machine (RVM) compiled byte code interpreter





Executed on mac mini with a 2.6GHz Quad-Core Intel Core i7 cpu, 16GB memory

Caveats

- Compiler is prepreprepreprepre-alpha
- This is the first real program *ever* that's been run on the compiler
- Memory usage is huge :-((being worked on)
- NB: The RVM is still an interpreter
- Compilation to JVM byte code work in progress

Conclusions (?)

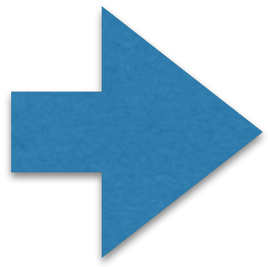
- It's too slow: parse+resolve+check soon is > 0.1 s
- Not *very* slow: up till 120k p+r+c is < 1 s
- Who has files of 250k anyway? ;-)
- Compiler has/will have significant impact
- Opportunities for incrementalization

Collaboration

or any other VCS…

Generic structure diff

```
form Foo {
  "What is your age?"
    age: integer

  "Another" y: integer

  if (age > 18) {

  }
}
```



```
form Bar {
  "Wat is je leeftijd?"
    age: integer

  "Another" x: integer

  if (age > 18) {
    "Yet another" z: integer
  }
}
```

```
diff {
  source = /input/version1.dql
  target = /input/version2.dql
  syntax = QL

  replace Id (5,3) with [[Bar]](5,3)

  replace Label (13,19) with [["Wat is je leeftijd?"]](13,21)

  delete Question 1 in (13,91)
  delete Question 2 in (13,91)

  insert Question [["Another" x: integer]](59,20) at 1 in (13,91)

  insert Question [[if (age > 18) {
      "Yet another" z: integer
    }
  ]](85,49) at 2 in (13,91)
}
```

source locations

language parametric

Todos

- Make a nice GUI for this

- Integrate it with Compare views built into Eclipse

Conclusions…

- “Reasonable” performance (?)
- Speed must, can and will be improved
- Collaboration via de facto standard VCS tools
- Opt-in: structural diffing (work in progress)