A #lang For All Seasons

Ben Greenman

1. Introduction

Racket is a programming-language programming language. It gives developers the tools to express and solve problems at the *language* level, rather than as formulas in a fixed logic, type system, or operational semantics. These language-building tools enable direct, concise solutions.

We demonstrate by example with the *iPoe* programming language: a system for **i**nteractive **po**etry **e**diting. On the surface, iPoe is a convenient program for writing poems in a known style, such as sonnets, limericks, and haikus. One level deeper, iPoe provides a specificiation language for defining new poetic styles. Our thesis is that students and practicing poets alike can benefit from this tool, or one like it, *despite having no knowledge of programming languages*. This is precisely because Racket lets us express the system at a high level of abstraction.

2. Example: Couplet

The following couplet is by Alexander Pope (Pope 1734). Prepending a language declaration `#lang ipoe/couplet` above the couplet makes a complete iPoe program.

```
Hope springs eternal in the human breast,
Man never is, but always to be blest.
```

Like all couplets of Pope's day, it is composed of two rhyming lines of iambic pentameter. The specification languge `ipoe/couplet` captures the essence of this poetic form:

```
#lang ipoe
#:name couplet
#:description "Two rhyming lines,
               10 syllables each."
#:rhyme-scheme {[ (A . 10)
                  (A . 10) ]}
```

As the in-program description states, a couplet is composed of:

- One stanza, delimited by square brackets
- Two lines, each delimited by parentheses
- A constraint on the rhyme and meter of each line, expressed as a symbol (A) and natural number (10).

The symbol `A` is a *rhyme variable*. When we match this specification against two lines of text, this rhyme variable is bound by the last word *w* of the first line. The second line is then constrained to end with a word that rhymes with *w*. In our example, the rhyme constraints translate to the proposition: "*breast* rhymes with *blest*".

The integer 10 used to encode syllables is more straightforward. It simply requires that the sum total of syllables in all words of the specified line is exactly 10. We compute syllables just as we compute rhymes: by asking a trusted internet authority[1] and caching the result in a local database.

[1] `http://rhymebrain.com/`

Unfortunately, we cannot express the finer constraint that Pope's couplets were written in iambic pentameter (meaning that 5 of the 10 syllables of each line are stressed, and the other 5 relatively silent). Nor can we express the opinion that this is a particularly good couplet. We leave these ideas to future work.

3. Example: Sonnet

While couplets are verse in their own right, they often appear as part of a larger poem. The English (or Shakespearian) Sonnet, for example, always ends with a couplet (Shakespeare 1608):

```
Shall I compare thee to a summer's day?
Thou art more lovely and more temperate:
Rough winds do shake the darling buds of May,
And summer's lease hath all too short a date:

Sometime too hot the eye of heaven shines,
And often is his gold complexion dimmed,
And every fair from fair sometime declines,
By chance, or nature's changing course untrimmed:

But thy eternal summer shall not fade,
Nor lose possession of that fair thou ow'st,
Nor shall death brag thou wander'st in his shade,
When in eternal lines to time thou grow'st,

So long as men can breathe, or eyes can see,
So long lives this, and this gives life to thee.
```

We can specify English Sonnets using the same techniques from the couplet specification. For brevity, however, we use the new directive `#:syllables` to declare that all lines, unless otherwise specified, should contain 10 syllables.

```
#lang ipoe

#:name english-sonnet
#:description "Five stanzas of iambic
  pentameter, followed by a couplet."
#:syllables 10
#:rhyme-scheme {
  [A B A B]
  [C D C D]
  [E F E F]
  [G G]
}
```

The rhyme scheme in this case specifies each line with a symbol rather than a pair. Secretly, these lines elaborate into full pairs. The first step of decoding a rhyme scheme will turn the first stanza `[A B A B]` into the list of pairs `[(A . *) (B . *) (A . *) (B . *)]`. On the other hand, giving only a number will leave an asterisk (wildcard) for the rhyme.

Also note that we did not use the specification of couplets directly in our specification of sonnets. This is another opportunity for future work.

4. Extra Constraints

These `rhyme-scheme` patterns are exactly the specification for many common poetic forms. Here are some classics:

```
;; clerihew
{[A A B B]}

;; haiku
{[5 7 5]}

;; limerick
{[(A . 9) (A . 9)
  (B . 6) (B . 6)
  (A . 9)]}
```

But some forms have additional structure. Here is a famous villanelle by Sylvia Plath (Plath 1953):

```
I shut my eyes and all the world drops dead;
I lift my lids and all is born again.
(I think I made you up inside my head.)

The stars go waltzing out in blue and red,
And arbitrary blackness gallops in:
I shut my eyes and all the world drops dead.

I dreamed that you bewitched me into bed
And sung me moon-struck, kissed me quite insane.
(I think I made you up inside my head.)

God topples from the sky, hell's fires fade:
Exit seraphim and Satan's men:
I shut my eyes and all the world drops dead.

I fancied you'd return the way you said,
But I grow old and I forget your name.
(I think I made you up inside my head.)

I should have loved a thunderbird instead;
At least when spring comes they roar back again.
I shut my eyes and all the world drops dead.
(I think I made you up inside my head.)
```

Using the tools developed so far, we can give a partial specification for this and other villanelles:

```
#lang ipoe

#:name villanelle
#:description "Six passionate triplets
               with a haunting refrain"
#:syllables 10
#:rhyme-scheme {
  [R1 B  R2]
  [A  B  R1]
  [A  B  R2]
  [A  B  R1]
  [A  B  R2]
  [R1 B  R2]
}
```

This specification misses, however, the constraint that all `R1` lines contain exactly the same words ("I close my eyes and all the world drops dead"). Same for all `R2` lines. We can express these and other positional equality constraints by calls to an API of zero-indexed selector functions. Here the the constraint for the first refrain:

```
#:constraint
  (line=? (line 0 (stanza 0))
          (line 2 (stanza 1))
          (line 2 (stanza 3))
          (line 0 (stanza 5)))
```

The real benefit of these constraints is that, once encoded, they make it very easy to check new poems or drafts against complicated specifications. Villanelles are not too difficult to check by hand, but other forms are quite complicated. For example, the first two stanzas of Seamus Heaney's *Two Lorries* (Heaney 1996) demonstrate the *sestina's* word-cycling structure:

```
It's raining on black coal and warm wet ashes.
There are tyre-marks in the yard, Agnew's old lorry
Has all its cribs down and Agnew the coalman
With his Belfast accent's sweet-talking my mother.
Would she ever go to a film in Magherafelt?
But it's raining and he still has half the load

To deliver farther on. This time the load
Our coal came from was silk-black, so the ashes
Will be the silkiest white. The Magherafelt
(Via Toomebridge) bus goes by. The half-stripped lorry
With its emptied, folded coal-bags moves my mother:
The tasty ways of a leather-aproned coalman!
```

Here, the constraint should include terms like:

```
(word=? (last-word (line 2 (stanza 0)))
        (last-word (line 5 (stanza 1))))
```

These index constraints are simple, but tedious to check by hand.

5. Family of Languages

All our poetic forms have been written in the language `#lang ipoe`. Our poems, in contrast, have been written in the more specific `#lang ipoe/couplet`, `#lang ipoe/english-sonnet`, and `ipoe/villanelle`. These specific languages are actually generated by programs written in `#lang ipoe`, which is a language for generating a family of little languages.

In full detail, programs written in `#lang ipoe` are interpreted as poem specifications. These specifications list critical data and metadata about a poetic form. Nothing more. The language `ipoe` then elaborates a specification into a new Racket language that implements the mechanical task of reading text from an input source and checking this new text against the rhyme scheme. Additionally, elaborated programs include a pass to check spelling, and also support a variety of run-time flags to control details of the poem-checking process.

Leveraging `#lang ipoe` as a language-generator makes it easy to define new poetic forms. The process takes 3 steps. First create the folders `portrait/lang` anywhere on the filesystem. Second, create a file `portrait/lang/reader.rkt` and fill it with:

```
#lang ipoe

#:name portrait
#:description "4-lines, biographical"
#:rhyme-scheme {[A B C B]}
#:syllables 8
```

Finally, run the command `raco pkg install ./portrait` to install the language. Now we can check and approve poems like the following (Joyce 1916):

```
#lang portrait

Stephen Daedelus is my name
Ireland is my nation
Clongowes is my dwelling place
```

```
And heaven my expectation
```

Wonderful! Now, the reason it is `#lang portrait` and not `#lang ipoe/portrait` is that we created and installed a new Racket package. If we prefer to use `#lang ipoe/portrait`, we must:

- Find the `ipoe/` folder on our filesystem.
- Move the folder `./portrait` into the `ipoe/` folder.
- Run `raco pkg update ipoe/` to compile the `portrait` reader.

Either way, it is easy to define new kinds of poems.

6. Discussion

A rhyme scheme is really a form of type specification for poems. The language `#lang ipoe` is a language for declaring a type system, and the `#lang ipoe/*` family are type-checked programs. It so happens that the syntax of these "programs" and semantics of type checking is simpler than what is found in many programming languages, but the core ideas are the same.

With this in mind, we hope to encourage others to explore tools for computer-aided composition that leverage concepts from programming languages to help students and professionals in other disciplines.

Finally, although we have argued that Racket is the ideal language to implement such tools, a critic might argue that any general-purpose language could do the same. After all, Java, C, and Haskell are all Turing complete, and at least Haskell has many powerful libraries for extending the core language. Indeed (Scholtes 2007):

```
Each computer, in theory, is suitable
To attack any problem computable.
This thesis (Church Turing),
Unproven, alluring,
Remains, as we speak, irrefutable
```

Point taken, but in closing we would like to remind readers that:

```
The gap between theory and practice
is, quite frequently, drastic.
And the programming language
can help you or hang you --
that's why I prefer hacking in Racket
```

Bibliography

Seamus Heaney. Two Lorries. 1996. `http://www.ppu.org.uk/learn/poetry/poetry_otherwars3.html` Accessed 2015-08-25

James Joyce. *Portrait of the Artist as a Young Man*. B.W. Huebsch, 1916.

Sylvia Plath. Mad Girl's Love Song. 1953. `https://en.wikipedia.org/wiki/Mad_Girl's_Love_Song` Accessed 2015-08-16

Alexander Pope. *An Essay on Man: Epistle I*. 1734. `http://www.poetryfoundation.org/poem/174165` Accessed 2015-09-29

Mike Scholtes. Church-Turing thesis. The Omnificent English Dictionary in Limerick Form, 2007. `http://www.oedilf.com/db/Lim.php?Word=Church-Turing%20thesis` Accessed 2015-09-27

Shakespeare. Sonnet #18. 1608. `https://en.wikipedia.org/wiki/Sonnet_18` Accessed 2015-08-16