PRASCAL: a pataphysical programming language

Andrew Hugill* and Hongji Yang

Creative Computing Cluster (CCC),
Bath Spa University,
Corsham Court, Corsham,
SN13 0BZ, England, UK
Email: a.hugill@bathspa.ac.uk
Email: h.yang@bathspa.ac.uk
*Corresponding author

Abstract: This paper introduces PRASCAL, a programming language that distorts traditional PASCAL using pataphysical principles. The aim of the language is to stimulate creativity and to embed playfulness in computer systems. A wider aim is to reach towards a less severe, more human, form of logic. Pataphysics was a concept elaborated by the French writer and poet Alfred Jarry (1873–1907) in a series of plays and novels, as well as through his own life. It is defined as the science of imaginary solutions and the science of the laws governing exceptions and contradictions. PRASCAL applies this concept through mechanisms such as patadata and Uboolean logic to arrive at a language which is always exceptional and particular.

Keywords: programming language; pataphysics; many-valued logic; creativity; play.

Reference to this paper should be made as follows: Hugill, A. and Yang, H. (xxxx) 'PRASCAL: a pataphysical programming language', *Int. J. Creative Computing*, Vol. X, No. Y, pp.xxx–xxx.

Biographical notes: Andrew Hugill is a composer and Computer Scientist, musicologist and literary scholar. He is the Director of the Centre for Creative Computing at Bath Spa University. His recent books include *The Digital Musician* (Routledge 2012) and *'Pataphysics: A Useless Guide* (MIT Press, 2012). His recent compositions include the installation *Secret Garden*, an 11-iPad opera/ballet created with Martin Rieser, which was experienced by more than 36,000 people during its recent exhibition in the Museum of Contemporary Art, Taipei.

Hongji Yang is the Deputy Director of the Centre for Creative Computing at Bath Spa University. He obtained his PhD degree at Durham University, England. His research interests cover computer organisation, networking, software engineering and recently creative computing, and he has published in all these areas (five books and well over 300 refereed papers). He was the Deputy Technical Director for the Software Technology Research Laboratory at DeMontfort University. He has been an organiser for several leading international conferences, such as the IEEE International Conference on Software Maintenance (1999) and the IEEE Computer Software and Application Conference (2002). He is a recipient of IEEE Computer Society Meritorious Service Award and Outstanding Contribution Award. He became a Golden Core Member of IEEE Computer Society since 2010.

1 Introduction

For creative computing to be truly creative, it must be surprising, playful, inventive, even humorous. The link between play and creativity is by now so well established as to have become an accepted fact. Huizinga (2001) argued the case persuasively in the 1940s, and Caillois (2001) continued the theme in the 1960s. Nachmanovitch (1990, p.42) summed up the main argument in 1990:

> Improvisation, composition, writing, painting, theater, invention, all creative acts are forms of play, the starting place of creativity in the human growth cycle, and one of the great primal life functions. Without play, learning and evolution are impossible. Play is the taproot from which original art springs; it is the raw stuff that the artist channels and organizes with all his learning and technique.

For software engineers, the notion of creative play has no less potential than for artists. Indeed, we have argued that writing software and making art are fundamentally similar activities (Hugill and Yang, 2013). It is perfectly possible to be highly creative in code, to display 'learning and technique', to improvise, compose, and write, even to paint and dramatise. Embedding this sense of creative play at the engineering level is therefore a key mission of creative computing. From a more scientific point of view, increasing our collective understanding of how creative play may be retained in the logic-driven world of computing is a fundamental ambition.

By way of example, we present below our recent efforts towards these ends. PRASCAL is a pataphysical programming language for creative computing. It is a partial solution to the problem of engineering a pre-logical state that stimulates creativity. It begins from an assumption that pataphysics offers an effective way to stimulate creativity (Hugill et al., 2013). The evidence supporting this assumption is extensive, since many of the most distinctively creative people have called themselves pataphysicians or explored pataphysical ideas in their work (Hugill, 2012). For instance, some of the most original thinkers and artists from the postwar period have been card-carrying members of the Collège de 'Pataphysique, from Marcel Duchamp to M.C. Escher, from Jean Baudrillard to Umberto Eco, from Eugene Ionesco to the Marx Brothers. What these figures have in common is a delight in playing with logic and language in ways which challenge the limits of reason.

Pataphysics began as a schoolboy joke at the Lycée de Rennes, France, in the 1880s. The target of the joke was the unfortunate science teacher, M. Hébert. One of the schoolboys was the writer Alfred Jarry (1873–1907) whose short life became largely devoted to the elaboration of a 'science' of pataphysics. In his hands, M. Hébert was transformed into the monstrous marionette-like figure Ubu, the main character in *UbuRoi*, a play whose première sparked riots and kick-started modernism (Shattuck, 1968). Jarry went on to develop increasingly sophisticated explanations of pataphysics, culminating in the posthumously published 1911 novel *Les Gestes et opiniens du Docteur Faustroll*, *pataphysicien* (Exploits and Opinions of Doctor Faustroll, pataphysician). Therein, pataphysics was defined as 'the science of imaginary solutions' and "the science of the laws governing exceptions and contradictions" [Jarry, (1996), p.21]. The word pataphysics is sometimes preceded by an apostrophe, thus: 'pataphysics [Hugill, (2012), pp.7–8].

2 Pataphysical creativity

Pataphysical knowledge consists of exceptions, contradictions, particularities and imaginary solutions. Its only generalisable proposition is that it has no generalisable propositions because, as Jarry (1996, p.22) wrote, it overturns the foundations of empirical science:

> Contemporary science is founded upon the principle of induction: most people have seen a certain phenomenon precede or follow some other phenomenon most often, and conclude there from that it will ever be thus. Apart from other considerations, this is true only in the majority of cases, depends upon the point of view, and is codified only for convenience – if that!

Since pataphysics exists beyond metaphysics[1], it also represents a meta-abstraction. Pataphysical knowledge is immediately useless, yet its continuous spiral motion provides an energy that is useful at the spiritual level. It subverts reason without being gibberish or merely silly. It uses the apparatus of science, art, mathematics and religion, but to ends that are rather different from those disciplines. Its products constitute *evidence* of pataphysics, rather than being pataphysics in themselves. It is entirely subjective and its solutions are all imaginary. It is important that any representation of this knowledge captures its playful and anarchic nature in a way that does not contradict its fundamental seriousness.

We can formulate a key question of pataphysical creativity as follows: how may we think illogically? Pataphysics generally agrees with Friedrich Nietzsche that '*pure logic* is the impossibility that grounds science'. He argued that there can be no direct correlation between the absolute knowledge of logical certainty and the ever-changing world of real things.

> One of those things that may drive a thinker into despair is the recognition of the fact that the illogical is necessary for man, and that out of the illogical comes much that is good. It is so firmly rooted in the passions, in language, in art, in religion, and generally in everything that gives value to life, that it cannot be withdrawn without thereby hopelessly injuring these beautiful things. It is only the all too naive people who can believe that the nature of man can be changed into a purely logical one; but if there were degrees of proximity to this goal, how many things would not have to be lost on this course! [Nietzsche, (2006a), p.33]

In the Nietzschean world-view, our idealised condition has lost the pervasive illogicality that preceded logic.[2] One way to liberate creativity is to try to recapture something of this illogicality. This vision seems to coincide with theories of creativity which regard 'tacit knowledge' as the originating source of the creative impulse (von Krogh et al., 2000). Tacit knowledge is often identified as an intangible human resource (Jacobson, 1990) or, as Polanyi (1967, p.4) famously wrote in *The Tacit Dimension*: "we can know more than we can tell". If creativity is located in the divergent thinking that taps into this unknowable store of pre-cognitive knowledge, then illogicality may provide a means to reach that state more readily.

The computational corollary to the question 'how may we think illogically' is: 'how may computers reason illogically'? While the aim of creative computing is not necessarily to produce a machine intelligence that is capable of creativity, it nevertheless does aim to enhance human creativity through the complementary development of intelligent machines (Hugill and Yang, 2013). Computing struggles with illogicality, because the need for structured reasoning drives the algorithmic process itself. Standard solutions such as fuzzy logic introduce a degree of controlled variability into the reasoning process (Tamir et al., 2015). The balanced ternary places the zero sum on a sliding scale between 1 and –1, which is a way of introducing a certain degree of unpredictability into processes which nevertheless still accord with the fundamental principles of von Neumann binary architecture. Such solutions, however, are incomplete in terms of pataphysical operations such as clinamen (inexplicable swerves), syzygy (unexpected alignments) and antinomy (co-existent contradictions).

This article describes only a partial solution to the problem of capturing the pre-logical state, because the computer, as presently configured, is incapable of operating illogically. The despised ideal state of scientific reasoning and absolute logic is incarnated in computers above all other physical products of mankind. The best we may achieve at this time is to subvert its logical processes in order to deliver pataphysical results. These are expressed as particularities, exceptions, contradictions and imaginary solutions. In future, a pataphysical compiler may be created that is capable of 'thinking' pataphysically. Perhaps quantum computing, with its inherent unpredictability, will offer the best platform for such an evolution. In the meantime, this paper will outline a new programming language which produces pataphysical results when implementing syzygistic, clinamenic and antinomial principles, representations and operations within a conventional logical format.

3 PRASCAL

PRASCAL combines artistic and scientific creativity within the context of the traditional programming language PASCAL, many of whose components remain unchanged. The pataphysical interventions within the PASCAL structs represent clinamen, syzygies and antinomies which enable results that are both humorous and creative [Hugill, (2012), pp.9–12]. The name 'PRASCAL' is itself an example of this, echoing Jarry's addition of the letter 'r' to the word *merde* to create the celebrated opening word of the play *Ubu Roi*: 'merdre'. This clinamen 'r' changes the meaning of the word and gives it added emphasis. The addition of an 'r' introduces a 'rascal' into PASCAL, which seems appropriate given the creatively playful nature of this language.[3]

We have selected PASCAL because it is a traditional language. Neither object-oriented languages nor aspect-oriented languages are suitable for this project. This is because we are only seeking to demonstrate how the new logic is used and how pataphysical operations are carried out. This will give us a complete language which can be compiled and interpreted by a computer. The main body of this paper therefore follows the standard pattern for PASCAL, except that we indicate only those new elements introduced by PRASCAL. In some instances, we may also give the context to aid comprehension. The primary aim of PRASCAL is to demonstrate programming for creative computing. This may also prepare for the arrival of quantum computing by making a language that is capable of creatively exploiting the ambiguities inherent in the qubit.

4 PASCAL/PRASCAL

We indicate below only those extensions to the standard PASCAL set (PASCAL Standard, 1991). A PRASCAL program has the same structure as its equivalent in PASCAL, that is to say:

- program name
- uses command
- type declarations
- constant declarations
- variables declarations
- functions declarations
- procedures declarations
- main program block
- statements and expressions within each block
- comments.

PRASCAL handles statements, loops, functions, strings, arrays, records and sets in exactly the same way as PASCAL. In other words, decision-making in PRASCAL is identical to PASCAL. File handling is also the same. The principal differences are therefore the extensions or distortions of the traditional language detailed below.

5 Crossing levels of abstraction

Crossing levels of abstraction has been identified as key element of creative computing [Hugill and Yang, (2013), p.14]. Colburn and Shute (2007) propose that the distinction between abstraction in mathematics and abstraction in computing lies in the fact that in mathematics abstraction is information neglect, whereas in computing it is information hiding. When we consider artistic creation, however, the word ‘abstraction’ has a somewhat different meaning. Pataphysical principles can help to resolve such disparities by offering a series of operators that will allow effortless movement between levels of abstraction in all domains.

PRASCAL introduces some new operators that allow traversal of levels of abstraction. These are the standard mathematical operators +, -, * /, preceded by an apostrophe. The apostrophe in pataphysics is a much-debated and elusive sign that seems to embody the idea of pataphysics itself. By placing it before the word ‘pataphysics (something he did on only a single occasion) Jarry (1996, p.21) avoided ‘a simple pun’. Since he never explained what the pun might be, speculation has raged ever since about the precise meaning of the apostrophe. But it has the merit of wrong-footing the reader in a playful and mysterious way, as well as symbolising the desire for the exceptional by trying to self-exclude the word from the dictionary. In PRASCAL, the apostrophe is generally used to denote the pataphysical element that may be used to influence an operation, function, variable, and so on.

Since pataphysics extends as far beyond metaphysics as metaphysics extends beyond physics we may express the implied crossing of levels grammatically as follows:

< Physics >'+ < Physics >::= Metaphysics (F1)

< Metaphysics >'+ < Metaphysics >::= Pataphysics (F2)

< Pataphysics >'– < Metaphysics >::= Metaphysics (F3)

< Metaphysics >'– < Physics >::= Physics (F4)

< Physics >'* < Metaphysics >::= Pataphysics (F5)

< Pataphysics >'/ < Metaphysics >::= Physics (F6)

6 Key words

PRASCAL statements contain some additional reserved words to the standard PASCAL set, for example:

- *absolu* (string) → string (the absolu value will always result in a cliché)[4]
- *antinomy* (string) → string (the antinomy function is self-contradictory)
- *clinamen* (string) → string (the clinamen function is a small swerve or deviation)
- *debraining* (string) → string (a debraining severs the head from the body or, in the case of a string, separates some of the initial letters from the rest)
- *epiphenomenon* (cstring) → string (an epiphenomenon is a by-product)
- *exception* (string) → string (the exception function is something outside the given category)
- *haha* (int) → int (haha is any number, the more absurd or unlikely the better)
- *particular* (string) → string (a particular is a unique precision)
- *pschitt* (string) → ubool (see below for an explanation of ubool)
- *spiral* (string) → string (the spiral variable is somewhat wild)
- *syzygy* (string) → string (the syzygy function is an unexpected collision)

The PRASCAL character set extends the standard PASCAL set as follows: '+, '–, '*, '/, ⬒, ◄, ≡, :–).

PRASCAL also adds two new data types to the classic Booelan, character, real and integer, namely: patadata and Uboolean.

7 Patadata

The concept of patadata has already been elaborated by Hendler and Hugill (2013, p.27):

> In thus envisaging a layer of meta-metadata, we will benefit from the ambiguities thrown up by the process of metadata creation, precisely those 'shortcomings' identified by Doctorow (2001) in his 'Metacrap' thesis: lying, laziness, error, subjectivity, plurality, and so on. But we will also be able to insert our pataphysical declensions into the metadata harvest in a way that will enrich the creative interaction, rather than just deliver a series of obvious mistakes.

The purpose of a data type is for the compiler to be able to reserve enough space in memory for that piece of data. PRASCAL extends the usual character (Char) to include Meta-Char and Pata-Char, where Meta-Char is a concept which is normally represented by two words, and Pata-Char is a super concept that is represented by up to four words. Therefore, two bytes are reserved for a Char type; four bytes for a Meta-Char; and eight bytes for a Pata-Char.

8 Uboolean data

An Uboolean type will take the same amount of memory space as a Boolean type. Uboolean is in effect a new kind of logic for computing that uses the antinomy of pataphysics. We call this Uboolean logic after King Ubu, whose vastly inflated belly is capable of accommodating – nay, comfortable with assimilating – the simultaneous existence of mutually exclusive opposites. In *César-Antichrist* (1895) Ubu's 'physick-stick' revolves across the stage, creating a minus and a plus sign as it does so. This double existence becomes one of the foundational principles of pataphysics itself. Whereas physical science relies on the repeated experiment, pataphysical science acknowledges the uniqueness of each and every individual moment or event. As Georges Perec observed:

> If physics proposes: 'you have a brother and he likes cheese', then metaphysics replies: 'If you have a brother, he likes cheese'. But 'Pataphysics says: You don't have a brother and he likes cheese.

9 Uboolean logic

In traditional Boolean logic, there are two possible states: true (T) and false (F). Uboolean logic allows a third state, the FalsTrue (FT), in which something is simultaneously false and true.[5] This rather dialetheistic form of logic not only produces pataphysical outcomes from traditional operators, but also new pataphysical operators that enable us to move between levels of abstraction. The following truth tables present the consequences of Uboolean logic. Consider the outcome *z* when the basic Boolean operations conjunction (and), disjunction (or) and negation (not) are applied to *x* and *y*:

Table 1 Truth table for *And* (∧)

X	*Y*	*Z = X (AND ∧)Y*
F	F	*F*
F	T	*F*
F	FT	*F*
T	F	*F*
T	T	*T*
T	FT	*FT*
FT	F	*F*
FT	T	*FT*
FT	FT	*FT*

Table 2 Truth table for *OR* (∨)

X	*Y*	*Z = X (OR ∨) Y*
F	F	*F*
F	T	*T*
F	FT	*FT*
T	F	*T*
T	T	*T*
T	FT	*T*
FT	F	*FT*
FT	T	*T*
FT	FT	*FT*

Table 3 Truth table for NOT (¬)

X	*Z = (NOT ¬) X*
F	T
T	F
FT	FT

Consider also the exclusive disjunction XOR ⊕.

Table 4 Truth table for XOR (⊕)

X	*Y*	*Z = X XOR ⊕ Y*
F	F	*F*
F	T	*T*
F	FT	*FT*
T	F	*T*
T	T	*F*
T	FT	*FT*
FT	F	*FT*
FT	T	*FT*
FT	FT	*T*

Uboolean bears a superficial resemblance to existing systems of many-valued logics. On closer inspection, however, there are some important differences. All previous many-valued logics concern an indeterminate value that is sometimes called the 'included middle'. Thus, Jan Lukasiewicz's classic paper of 1930 introducing his three-valued logic system, describes an intermediate truth-value I.

> "I can assume without contradiction that my presence in Warsaw at a certain moment of time next year, e.g., at noon on 21st December, is not settled at the present moment either positively or negatively. It is therefore possible but not necessary that I shall be present in Warsaw at the stated time. On this presupposition the statement "I shall be present in Warsaw at noon on 21st December next year" is neither true nor false at the present moment" [Kneale and Kneale, (1962), p.570].

D.A. Bochvar's ontological system, developed in 1939, similarly treats I as 'undecided' [Rescher, (2007), p.66], whereas Kleene's (1952, p.332) later epistemological system refers to I as 'unknown'. There have been variants on this principle. The Chinese logician Moh Shaw-kwei, for example, "proposed that the intermediate truth-value I be construed as 'paradoxical' and be assigned to propositions like 'This statement is false' which will be false if classed as true and true if classed as false" [Rescher, (2007), p.70]. More recently, Bergstra and Ponse (1998, p.95) have proposed that the value D should be substituted for I to express divergence. This takes us somewhat closer to pataphysical ideas, but a close perusal of their system shows that D and I are nevertheless fairly interchangeable and remain indeterminate. Finally, the m-valued systems of Emil Post dispense with the true/false distinction altogether, but only to effect "a cyclic shift in the truth-values" [Rescher, (2007), p.76].

Uboolean logic, by contrast, describes the simultaneous existence of mutually exclusive opposites. The FalsTrue is not some indeterminate, undecided, divided, or even paradoxical, instance of an included middle. Rather it is an explosive combination of irreconcilable states of affairs whose only possible outcome is an imaginary solution. This cannot be instantiated, because there is no generalisable example of a standard imaginary solution arising from a FalsTrue condition. Any such example is by definition exceptional. Herein lies the crucial distinction: uboolean logic aims to stimulate creativity through pataphysical knowledge. In other words, it is inherently illogical, subjective and ungeneralisable, in short: human. As Robin Ahlgren recently commented: "We really need a step sideways in logic so that we can think more humanly and in less black and white terms. Jarry's 'correspondence' with the scientists of his day seems to reach out for a more natural, less severe form of science." (Ahlgren, private message to author, 9.3.16).

Many-valued logics have already been applied in computing as nonbinary-valued systems. The comprehensive survey by Miller and Thornton (2007) shows their implementation in hardware and software applications, and includes their use in quantum computing. Porat (1969, p.947) argues that "a system which is based on a radix higher than 2 and built from multivalued elements offers the advantages of: (a) higher speed of serial and some serial-parallel arithmetical operations […]; and (b) better utilisation of transmission channels […]; and (c) more efficient error detection and correction codes; and (d) potentially higher density of information storage". The implementation of Uboolean logic, on the other hand, makes no such claims to increased usefulness. Rather, its main purpose is to represent the apostrophe before the word 'pataphysics by wrong-footing conventional logics. Its humour is embedded in its construction, such that the playful nature of the resulting outputs is matched only by the absurdity of the underlying propositions.

Type Declarations follow the PASCAL convention: type-identifier-1, type-identfier-2 = type-specifier; for example:

```
days, age = integer;
yes, true = boolean;
name, city = string;
fees, expenses = real;
```

However, PRASCAL adds the following:

```
falstrue = uboolean.
```

10 Constants

A constant is an entity that remains unchanged during program execution. PRASCAL allows ordinal, set, char and string constants, but is primarily concerned with the string type.

```
const
Identifier = contant_value;
```

The following are some examples of constant declarations, in addition to existing PASCAL constants:

```
PI = 3.141592;
NAME = ‘Alfred Jarry’;
CHOICE = yes;
OPERATOR = ‘+’;
INFINITY = ∞
ABSOLUTE ZERO = ‘K’
```

11 Enumerated types

```
type
enum-identifier = (item1, item2, item3, ... )
```

```
PATAPHYSICIANS = (Faustroll, Ubu, Sandomir, Sainmont, Mollet, Opach,
Lutembi)
SPIRALS = (Gidouille, Archimedean, Euler, Fermat, Hyperbolic, Lituus,
Logarithmic, Fibonacci)
   CONVEYANCE = (Bicycle, Time Machine, Carapace, Skiff)
```

12 Subrange types

var

age: '63 … 63; Here, ‘63 represents the imaginary birth age of Dr. Faustroll, where the apostrophe elides his years 1 - 62, which never existed.

type

subrange-identifier = lower-limit ... upper-limit;

```
const
P = '63;
Q = 63;
type
Number = '63 ... 63;
Value = P ... Q;
```

type

Constraints

```
Anoulipism
Snowball
Heterogram
Homomorphism
S+7
ElementaryMorality
Oulipism
Combinatorics
Quenine
Synthoulipism
```

13 Variable types

To the standard variable types (character, integer, Boolean, enumerated, subrange and string), we add Uboolean.

Variable declarations:

var

variable_list : type;

var

age, weekdays : integer;

taxrate, net_income: real;

choice, isready: boolean;

initials, grade: char;

```
name, surname : string;
imagine, solves: Uboolean
Enumerated Variables:
var
var1, var2, ... : enum-identifier;
type
months = (Absolu, Haha, As, Sable, Décervelage, Gueules, Pédale, Clinamen, Palotin,
Merdre, Gidouille, Tatane, Phalle);
Var
m: months;
...
M := Absolu;
Subrange Variables:
var
subrange-name : lowerlim ... uperlim;
var
clinamen: K … ∞;
age: ‘63 ... 63;
```

14 Operators

In PRASCAL, operators mainly operate on variable strings. PRASCAL introduces several new special operators to complement the standard PASCAL sets. Boolean and bit operators remain the same (although PRASCAL does not use bit operators).

In the following tables: A and B are physical entities; ∞ is metaphysical infinity; im is imaginary; and ∞ – a – 0 + a + 0 is the formula by which Dr. Faustroll demonstrates that God is a point.

Table 5 Arithmetical special operators

Operator	*Description*	*Example*
'+	Pataphysically adds two operands	A '+ B will give ∞
		∞ '+ ∞ will give ∞ – a – 0 + a + 0
'–	Pataphysically subtracts two operands	∞ – a – 0 + a + 0 '– ∞ will give A
		∞ '– A will give B
'*	Pataphysically multiplies two operands	A '* ∞ will give ∞ – a – 0 + a + 0
'/	Pataphysically divides two operands	'∞ – a – 0 + a + 0 ’/ ∞ will give A

Table 6 Relational special operators

Operator	*Description*	*Example*
□	Checks if the values of two operands are mutually exclusive. If yes, then condition becomes true.	(A = Ā) is true.
◀	Checks if the value of right operand may be imagined from value of left operand. If yes, condition becomes true.	(A = im)
≡	Checks if the value of left operand is equivalent, but not equal, to right operand. If yes, then condition becomes true.	(A ≡ A') is true.
:-)	Checks if the value of left operand stands in a funny relationship to value of right operand. If yes, then condition becomes true.	(A :-) A') is true.

The new pataphysical operators take precedence over all other operators.

Table 7 Prascal operator precedences

Operator	*Precedence*
'*, '/, '+, '–	Highest
~, not,	
*, /, div, mod, and, &	
|, !, +, -, or,	
=, <>, <, <=, >, >=, in	
or else, and then	Lowest

The following are pataphysical operations:

- ⊤ ::= EXCEPT (an exception).
- ¬ ::= CONTRA (a logical incompatibility between two or more propositions or statements that are simultaneously affirmed (i.e., both are true). CONTRA always negates the part of a string that PRECEDES the operator. e.g., I have a brother CONTRA he likes cheese, = I do not have a brother and he likes cheese.
- *im* ::= IMAGINE – designed to encourage hypothetical speculations or imaginary solutions.
- :-) = HAHA – humour or irony.
- ≡ ::= EQUIV – the pataphysical law of equivalence can take the form <variable> EQUIV <variable> where anything at all may be a variable. Example from Lewis Carroll: cabbages EQUIV kings.
- σ ::= EPI – superinduced upon, as in Jarry – 'An epiphenomenon is superinduced upon a phenomenon'.[6]

15 Grammar

The grammar of PRASCAL may be illustrated by the following syntactical examples, notated in Backus-Naur form (BNF), which we may rename Bosse-de-Nage Form, after the dog-faced baboon who accompanies Dr. Faustroll and only ever says ‘ha ha’. Some of the illustrative operations are taken from the work of the Oulipo, a workshop for potential literature whose members included several mathematicians and computer scientists (Motte, 2015; Mathews and Brotchie, 2005). These include the ‘snowball’, a poetic technique in which a letter is added or subtracted line by line, and S+7, a word substitution technique in which substantives are replaced by the seventh following substantive in a chosen dictionary. For example:

```
<clinamen>
    <clinamen> ::= <wordplay> | <semantic>
        <wordplay> ::= <snowball> | <playfair> | <S+7>
            <snowball> ::= <addletter> | {<addletter>} | <subtractletter> |
            {<subtractletter>} //<addletter> adds a letter to a word (at the end, the
            beginning or in the middle) to create a new word.
            <playfair> ::= <letterpair>* //<playfair> is a cipher method by which a word
            is broken up into pairs of letters.
            <S+7> ::= <substantive> <substantive+1> <substantive+2> <substantive+3>
            <substantive+4> <substantive+5> <substantive+6> substantive+7 <EOL>
            //A substantive may be a noun, verb, adjective or adverb.
        <semantic> ::= <synonym> | <association> | <dissociation> //Synonym is a
        10% deviation. Association is a 50% deviation, using a word association lookup
        such as http://wordassociations.net/ Dissociation is a 100% deviation, which
        works by finding a word that, when additively combined with the entered word,
        delivers 0 hits from a web search. Example: ragwort+stuplime.
<syzygy>
    <syzygy> ::= {<concept> | <thing> | <pun>}*2
        <concept> ::= <abstraction> | <plan> | <intention>
        <thing> ::= <object> | <entity> | <fact> | <creature>
        <pun> ::= (<word> <word>) <wordplay>
<antinomy>
    <antinomy> ::= <statement1> ¬ <statement2> // ¬ “contradicts” or “negates”.
```

16 Libraries

The libraries for PRASCAL are drawn from appropriately pataphysical sources. They include distorted collections of typical literary and mathematical formulae, such as clichés or number substitutions for values such as π. We also use Dr. Faustroll’s ‘Library of 27 Equivalent Books’. We may call a function such as: ReturnOneOfTheChosenFew(Random); this would return a value from the chosen few instances from the equivalent books, as follows:

1 From Baudelaire, E. A Poe’s Silence.

2 The precious tree into which the nightingale-king and his subjects were metamorphosed, in the land of the sun.

3 The Calumniator who carried Christ on to a high place.

4 The black pigs of Death, retinue of the Betrothed.

5 The ancient mariner’s crossbow and the ship’s floating skeleton, which, when placed in the skiff, was sieve upon sieve.

6 The diamond crowns of the Saint-Gothard rock-drillers.

7 The duck placed by the woodcutter at the children’s feet, and the 53 trees with scored barks.

8 The hares, running over the sheets, which became cupped hands and carried the spherical universe like a fruit.

9 Scapin’s lottery ticket.

10 The eye of the third Kalender, who was the son of a king: the eye poked out by the tail of the flying horse.

11 The 13 journeymen tailors massacred at dawn by Baron Mordax on the order of the knight of the papal order of Civil Merit, and the table napkin which he tied round his neck beforehand.

12 One of the golden peals from the celestial gold-smiths’ shops.

13 The scarab, beautiful as the trembling of hands in alcoholism, which vanished over the horizon.

14 The lights heard by the first blind sister.

15 The virgin, the bright, and the beautiful today.

16 The north wind which blew upon the green sea and blended with its salt the sweat of the galley slave who rowed until he was a 120 years old.

17 The joyful walk of the irreproachable son of Peleus in the meadow of asphodels.

18 The reflection, in the mirror of the shield silvered with ancestral ashes, of the sacrilegious massacre of the seven planets.

19 The little bells to which the devils danced during the tempest.

20 Cleopatra.

21 The sorrel plain where the modern centaur snorted.

22 The icicles hurled by the wind of God into the waters.

23 The scaly animals imitated by the whiteness of the leper’s hands.

24 r (the fifth letter of the first word of the first act of Ubu Roi).

25 The cross made by the spade in the horizon’s four brows.

26 Voices asymptotic toward death.

27 The two and a half leagues of the Earth’s crust.

We also use a library of pataphysical statements, so that a call such as ReturnAPataphysicalStatement(Random) might give one of the following:

1 Pataphysics is the end of ends (Sandomir).

2. Pataphysics is to metaphysics as metaphysics is to physics (Jarry).

3 Pataphysics is the science of imaginary solutions (Jarry).

4 Pataphysics is the science of the particular and the laws governing exceptions (Jarry).

5 Pataphysics describes a universe supplementary to this one (Jarry).

6 Whether it suits us or not, everything we do is ‘Pataphysics (Vian).

7 Beyond ‘Pataphysics lies nothing; ‘Pataphysics is the ultimate defense (Shattuck).

8 For pataphysics there’s no longer any singularity (Baudrillard).

9 I would never belong to any club that would have me as a member (Groucho Marx).

10 You do not have a brother and he likes cheese (Georges Perec).

These may be seen in operation in some of the examples below.

17 Examples

To clarify the above still further, consider the following examples. The first exemplifies the application of Uboolean logic in PRASCAL. The second exemplifies the ability of PRASCAL to cross levels of abstraction. The third exemplifies pataphysical functions.

Example 1: Wave-particle duality.

Given that Einstein, writing in 1905, concluded that light is both a wave and a particle (thereby contradicting both the Newtonian particle theory and the Maxwellian electromagnetic wave theory), we may test his conclusion using Uboolean logic.

Is light: having nothing to do with particle (NotP), particle plus something else (PPlus), or only particle (OnlyP)?

Is light: having nothing to do with wave (NotW), wave plus something else (WPlus), or only wave (OnlyW)?

Table 8 Wave-particle duality truth table

X (light with particle)	*Y (light with wave)*	*$Z = X \wedge Y$ (light is)*
F (NotP)	F (NotW)	*F*
F (NotP)	T (WPlus)	*F*
F (NotP)	FT (OnlyW)	*F*
T (PPlus)	F (NotW)	*F*
T (PPlus)	T (WPlus)	*T*
T (PPLus)	FT (OnlyW)	*FT*
FT (OnlyP)	F (NotW)	*F*
FT (OnlyP)	T (WPlus)	*FT*
FT (OnlyP)	FT (OnlyW)	*FT*

In PRASCAL, this would possibly be expressed as follows:

```
writeln('Is light: having nothing to do with particle (NotP), particle plus something
else (PPlus), or only particle (OnlyP)?');
Readln(AnswerP);
writeln('Is light: having nothing to do with wave (NotW), wave plus something else
(WPlus), or only wave (OnlyW)?');
Readln(AnswerW);

If (AnswerP = 'NotP') then LightP := F
If (AnswerP = 'PPlus') then LightP := T
If (AnswerP = 'OnlyP') then LightP := FT
If (AnswerW = 'NotW') then LightW := F
If (AnswerW = 'WPlus') then LightW := T
If (AnswerP = 'OnlyW') then LightW := FT
Light := LightP and LightW
If (Light = T) then writeln ('Pschitt');
If (Light = F) then writeln ('Fruck');
If (Light = FT) then writeln ('Ha ha');
```

Example 2: Crossing levels of abstraction

By applying formulae, F1 to F6 in a previous section, crossing levels of abstraction can be achieved as follows.

If we take a frog as an instance of physics and Heqet (the Ancient Egyptian frog goddess) as an instance of metaphysics, then a hoquet (hiccup, or antiphonal musical technique) is, by a clinamen (deviation), an instance of pataphysics. This gives us the following examples:

“frog” '+ “frog” = “heqet”

“heqet” '+ “heqet” = “hoquet”

“hoquet” '– “heqet” = “heqet”

“heqet” '– “frog” = “frog”

“frog” '* “heqet” = “hoquet”

“hoquet” '/ “heqet” = “frog”

More directly, we may line up the following representations: Human (physics), God (metaphysics), Dr. Faustroll (pataphysics), which gives the following:

“Human” '+ “Human” = “God”

“God” '+ “God” = “Faustroll”

“Faustroll” '– “God” = “God”

“God” '– “Human” = “Human”

“Human” '* “God” = “Faustroll”

“Faustroll” '/ “God” = “Human”

In natural language, this may be written, pataphysically, as:

A human added to a human produces God.
God added to God produces Faustroll.
God removed from Faustroll leaves God.
A human removed from God leaves a human.
A human multiplied by God gives Faustroll.
Faustroll divided by God results in a human.

Example 3: Pataphysical functions

- *absolu* (char_string) → string (the absolu value will always result in a cliché)

  Example: absolu(frog) = a frog in the hand is worth two in the refrigerator.

  (Note that this example draws upon a library of clichés, and that the ‘refrigerator’ substitution for the more predictable ‘pond’ is itself something of a clinamen swerve).

- *antinomy* (char_string) → string (the antinomy function is self-contradictory)

  Example: antinomy (brother) + antinomy (cheese) = You do nit have a brother and he likes cheese (cf. Perec).

- *clinamen* (char_string) → string (the clinamen function is a small swerve or deviation)

  Example: clinamen (rock) + clinamen (piano) + clinamen (Sisyphus) = billiards, limestone, roll.

  This example uses the 50% deviation syntax.

- *debraining* (char_string) → string (a debraining severs the head from the body or, in the case of a string, separates some of the initial letters from the rest)

  Example:

  debraining(chair) = hair

  debraining(hair) = air

- *epiphenomenon* (char_string) → string (an epiphenomenon is a by-product)

  Example: epiphenomenon (spaghetti) = steam

- *exception* (char_string) → string (the exception function is something outside the given category)

Example: exception (boats) = colander

- *haha* (int) → int (haha is any number, the more absurd or unlikely the better)

  Example: haha(0) = 37

- *particular* (char_string) → string (a particular is a unique precision)

  Example: particular (baboon) = Bosse-de-nage

- *pschitt* (char_string) → ubool (see below for an explanation of ubool)

  Example: pschitt (absolute) = there are no absolutes

- *spiral* (string) → string (the spiral variable is somewhat wild)

  Example: spiral (mermaid) = obeliscolychny

- *syzygy* (char_string) → string (the syzygy function is an unexpected collision)

  Example: syzygy (sister) + syzygy (tears) = having a cry, sis?

We can see from these examples that there are many potential applications of the PRASCAL language for stimulating creativity. In the 'clinamen' instance given above, the phrase 'billiards, limestone, roll' may well act as an inspiration to a new train of thought that could arise in a creative context. Nor need the examples be restricted to words alone: any of these could become inputs to a visual or multimedia search. The point is that the unexpected nature of the outcomes are not simply random but evocative and even poetic, offering somewhat divergent lines of thought by stimulating the imagination. Sometimes the results are exact, but more often they are ambiguous, or imprecise, reflecting the 'hit-and-miss' approach of creative thinking.

18 Concluding remarks

PRASCAL enables various pataphysical concepts to be expressed and manipulated. It has demonstrated how both humorous and scientific principles can co-exist in a programming language. Initial experiments with the language have shown clearly that a stimulus for creative thinking can be produced by the use of the language. The language may also be a starting point for exploring a new way of computer reasoning, computing creativity and even quantum computing.

References

Bergstra, J.A. and Ponse, A. (1998) 'Kleene's three-valued logic and process algebra', *Information Processing Letters*, Vol. 67, pp.95–103.

Caillois, R. (2001) *Man, Play and Games*, University of Illinois Press, Chicago.

Colburn, T. and Shute, G. (2007) 'Abstraction in computer science', *Minds and Machines*, Vol. 17, No. 2, pp.169–184.

Doctorow, C. (2001) *Metacrap: Putting the Torch to Seven Straw Men of the Meta-utopia* [online] http://www.well.com/~doctorow/metacrap.htm (accessed 25 March 2015).

Hendler, J. and Hugill, A. (2013) 'The syzygy surfer: (ab)using the semantic web to inspire creativity', *Int. J. Creative Computing*, Vol. 1, No. 1, pp.20–34.

Hugill, A. (2012) *Pataphysics: A Useless Guide*, MIT Press, Cambridge, MA.

Hugill, A. and Yang, H. (2013) 'The creative turn: new challenges for computing', *International Journal of Creative Computing*, Vol. 1, No. 1, pp.4–19.

Hugill, A., Yang, H., Raczinski, F. and Sawle, J. (2013) 'The pataphysics of creativity: developing a tool for creative search', *Digital Creativity*, Vol. 24, No. 3, pp.237–251.

Huizinga, J. (2001) *Homo ludens*, Routledge, London.

Jacobson, R. (1990) 'Unobservable effects and business performance', *Marketing Science*, Vol. 9, No. 1, pp.74–85.

Jarry, A. (1996) Translated S. Watson Taylor, *Exploits and Opinions of Doctor Faustroll, Pataphysician*, Exact Change, Boston.

Kleene S.C. (1952) *Introduction to Metamathematics*, Princeton, Amsterdam, pp.332–340.

Kneale, W. and Kneale, M. (1962) *The Development of Logic*, Clarendon Press, Oxford, pp.569–70).

Mathews, H. and Brotchie, A. (2005) *Oulipo Compendium*, Atlas Press, London.

Miller D.M. and Thornton, M.A. (2008) 'Multiple valued logic: concepts and representations', *Synthesis Lectures on Digital Circuits and Systems*, Vol. 12, pp.41–42.

Motte, W.F. (2015) *Oulipo – A Primer of Potential Literature*, Dalkey Archive Press, London.

Nachmanovitch, S. (1990) *Free Play: Improvisation in Life and Art*, Tarcher/Penguin, New York.

Nietzsche, F. (2006a) *Human, All Too Human*, Dover, New York.

Nietzsche, F. (2006b) *The Gay Science*, Dover, New York.

PASCAL Standard (1991) ISO/IEC 10206: Information Technology, Programming Languages, Extended Pascal.

Polanyi, M. (1967) *The Tacit Dimension*, Anchor Books, New York.

Porat, D.I. (1969) 'Three-valued digital systems', *Proceedings of the Institution of Electrical Engineers*, Vo. 116, No. 6, pp.947–954.

Rescher, N. (2007) 'Many-valued logic', *Topics in Philosophical Logic*, Vol. 17, pp.54–125.

Shattuck, R. (1968) *The Banquet Years: The Origins of the Avant-Garde in France 1885 – World War I*, Random House, New York.

Tamir, D.E., Rishe, N.D., Naphtali, D. and Kandel, A. (Eds.) (2015) *Fifty Years of Fuzzy Logic and its Applications*, Springer, London.

von Krogh, G., Ichijo, K. and Nonaka, I. (2000) *Enabling Knowledge Creation: How to Unlock the Mystery of Tacit Knowledge and Release the Power of Innovation*, Oxford University Press, Oxford.

Zenkin, A.A. (1997) 'Superinduction: a new method for proving general mathematical statements with a computer', *Doklady Mathematics*, Vol. 55, No. 3, pp.410–413.

Notes

1 "Pataphysics [extends] as far beyond metaphysics as the latter extends beyond physics" [Jarry, (1996), p.11].

2 "Where has logic originated in men's heads? Undoubtedly out of the illogical, the domain of which must originally have been immense" [Nietzsche, (2006b), p.84].

3 'Ill-posed software' posted a web page briefly outlining some ideas for pataphysical programming in 2006 which show a similarly playful aspiration. Regrettably, the project seems never to have been implemented http://www.illposed.com/philosophy/pataprogramming.html (accessed 2.08.15).

4 "Clichés are the armature of the absolute" (Jarry)

5 Note the phonetic (and nearly anagrammatic) echo in 'FalsTrue' of the French pronunciation of 'Faustroll'.

6 "Classical induction (according to J.S. Mill) is the process of making a *plausible* inference of a general statement from a *particular* statement. Therefore, it is quite reasonable to call the process of making a certain inference of a general statement from a *singular* statement *the method of superinduction*" [Zenkin, (1997), p.411]. A singular statement is any simple statement with a singular term as the subject, and which ascribes some property to the referent of the singular term, e.g., Ubu is King of Poland. The method of superinduction therefore give us the certain general statement: Poland is nowhere.