Revised$^{\infty}$ Report on the Useless Language Eightbol

Jeffrey Overbey and Brian Foote[1]

Dedicated to the Memory of Scheme

ABSTRACT

This report gives an authoritative description of the programming language Eightbol. According to the *Revised*[4] *Report on the Algorithmic Language Scheme,* "Programming languages should be designed not by piling feature on top of feature, but by removing the weaknesses and restrictions that make additional features appear necessary." Eightbol takes this philosophy to the extreme by eliminating behavior altogether. It is, to our knowledge, the first programming language which expressly forbids programs from doing anything.

General Terms
Languages

Keywords
Design by committee, programming languages, satire, semantics, standardization, theory, waste of time, XML

INTRODUCTION

Our fellow language designers, particularly in the academic community, have long sought to design languages more useless than their predecessors. Eightbol is the latest in a long line of languages of dubious utility, but it exceeds them all by doing absolutely nothing well. Our goal was to create an equally useless language but without the troublesome runtime semantics. By denying programs the ability to do anything, Eightbol not only achieves an unsurpassed level of uselessness; it also eliminates nearly all of the problems that have plagued computer scientitsts for decades. While some may be skeptical of the fact that Eightbol is not Turing-complete, we believe that Eightbol's simplicity and fashionable XML-based syntax more than compensate for this purported shortcoming. It is our hope that Eightbol's detractors will soon share our regard for runtime behavior as a needless and superfluous source of complexity.

ADVANTAGES

It has long been a dream of practicing programmers that any syntactically-correct program be provably correct and execute without error. However, avenues toward such guarantees have long eluded formal methodologists, who have found the requirement that programs actually do something to be an enduring source of frustration. We find the solution in Eightbol: Thanks to its innovative and unique total lack of behavior, theoreticians will find that proofs of program correctness are trivial, while practitioners can rejoice in the total elimination of the testing and maintenance phases of the software lifecycle, reducing project costs by an average of 82% [1].

SYNTAX AND TYPING

Correct Eightbol programs comprise a sequence of suggestions which the compiler is required to ignore. Suggestions may, however, be fully typed. The type system is implementation-dependent and shall be selected by the implementor according to the number of POPL papers it will generate.

In accordance with the industry's induction of XML as its Esperanto of structured data, Eightbol programs are expressed as XML documents. A Document Type Definition (DTD) for Eightbol follows.

```
<?xml version="1.0"?>
<!ELEMENT program (section)>
<!ELEMENT section (#PCDATA)>
  <!ATTLIST section name CDATA "(anonymous)">
```

This gives rise to the following "Hello, world!" program.

```
<?xml version="1.0"?>
<program>
  <section name="Hello, world!">
  </section>
</program>
```

[1]Although this is a work of satire, the first author will grudgingly accept (and subsequently ignore) correspondence sent to MC 258, 201 N Goodwin, University of Illinois at Urbana-Champaign, Urbana, IL 61801.

Future versions of the language may include a mechanism for extensible syntax. While some claim that this facility is overkill, citing the fact that Eightbol programs do absolutely nothing, the language designers conjecture that this facility will be well-received in academic circles, where it is commonly felt that the amount of work done should not place unnecessary limits on the amount of syntax used to describe that work. The subcommittee investigating this extension to the language will also consider augmenting the present Unicode character set with all 16×16-pixel bitmaps in order to placate theoreticians who, having exhausted all of the existing alphabets of the world, believe that cryptic, proprietary symbols will communicate their ideas most clearly and effectively. Such an option may be enabled in some prototype compilers via the `--brevity-fetish` command line argument.

SEMANTICS

Despite the prevalence of informal, English-language semantics in programming language specifications, the members of the Eightbol committee unanimously agreed that Eightbol is a canonical representative of the class of languages that can be effectively described by a formal denotational semantics.

Given an Eightbol program P, the behavior of P can be described by mapping it into the flat domain $\{\bot, 0\}$:

$$[\![P]\!] = 0.$$

IMPLEMENTATION

At the time of writing, the only known compiler for Eightbol [2], implemented as a front end for the GNU Compiler Collection, also includes a `print` suggestion, which, in flagrant disregard for Eightbol's required lack of behavior, allows the programmer to print messages to the screen. We cannot overemphasize that such behavior will not be tolerated. Noncompliant implementations should expect to be censured formally by the Eightbol standardization committee.

IMPACT

Eightbol can benefit the entire field of computing in ways that are almost too numerous to mention. Programmers will delight in the virtually nonexistent learning curve, finding that their first Eightbol programs compile and execute correctly on the first attempt (once the syntax errors have been removed). Educators will finally be able to eliminate debugging from the computer science curriculum, just as they have eradicated other archaic and useless topics, such as punch cards, PL/I, and math. To theoreticians, the single most important feature of Eightbol is its guarantee of semantic idempotence (or "impotence"), a property which makes the behavior of Eightbol programs easier to predict and analyze than the behavior of programs written in semantically potent languages. Moreover, it is ideal for modeling applications, since it is trivial to convert models to provably correct executable code. Compiler writers will finally be able to focus their efforts entirely on the elegantly mathematical aspects of parsing and type checking rather than on kludgy optimizations, heuristic parallelizers, and incomprehensible back ends. Computer architects can declare their field closed, since compiled Eightbol programs do not require complex, energy-inefficient microprocessors but can rather be executed equally well by ordinary objects, such as a rock or a spoon, simultaneously fulfilling the dreams of researchers in pervasive computing. Furthermore, no matter what execution platform is chosen, there are no limits on performance of the sort present in languages with behavior.

CONCLUSION

Eightbol is the pinnacle of language design by committee. The Eightbol committee was formed in October, 2005, with the mission to design a completely permissive and totally ineffectual language. Since that time, we have addressed the common complaints of nearly all groups potentially affected by the language, from compiler writers to formal methodologists to practicing programmers. The result is an unlikely eruption of XML and denotational semantics coupled with a novel lack of dynamism whose practical utility is surpassed only by its ability to preview forthcoming useless programming language research.

References

[1] Zelkowitz, M. V., Shaw, A. C., and Gannon, J. D.. *Principles of software engineering and design.* (Eaglewood Cliffs, NJ: Prentice-Hall, 1979).

[2] *The Eightbol project.* http://www.eightbol.org