Holy States Can Save the World!

Brother Jonathan Aldrich
High Monk of the Plaid Brotherhood
jonathan.aldrich@cs.cmu.edu

ABSTRACT

The twin evils of imperative and functional programming threaten the software world as we know it—destroying fields and spreading garbage throughout the cybersphere. In this paper we present Holy States as the last, best hope for peace and harmony. Holy States avoid the wanton destruction of field values so common in languages based on caffeine and out of tune notes. Neither is computation based on wasteful Schemes creating duplicate objects that then MilL around until they are garbage. Instead, every object is considered sacred and when no longer needed is reborn in a new, holy state. Recapturing the original stateful spirit of Turing's Machines, we show all computations can be expressed in a Holy way, with neither garbage nor field destruction.

Categories and Subject Descriptors

J.8.1 [**Computer Applications**]: Theology—*Saving the World*

General Terms

Languages, Human Factors.

Keywords

assignment, garbage, states, holiness, salvation.

ALABAMA. INTRODUCTION

Imperative programming is evil. With every "assignment" executed, some poor variable value is executed as well. And be not deceived by the colloquial use of "execution" in the field of computer science—it's true meaning is that found in the dictionary: the infliction of capital punishment [7]. Values are slaughtered mercilessly. The carnage must stop, before the world is destroyed!

Functional programming [6] was developed as an alternative to imperative programming, but it only substitutes one evil for another. Data structures are created willy nilly, to be used a few times, perhaps only once, before being discarded as "garbage." It is well-known that functional programming leads to a truly scandalous waste, contributing to the well-know problems of cycle depletion [16]. While efforts have been made to construct so-called "garbage collectors," even ones that integrate composting [8] this is clearly a misguided attempt to patch things up after the damage has been done. We must avoid the creation of garbage in the first place, or else the world will be suffocated by the reeking stench created by functional computation.

In this paper, we show how a hallowed approach to *typestate-oriented programming*[1] can save the world from the ravages of imperative assignment and the wasteland of functional garbage. To do so, we reach back into the mists of time to recapture the true intent of the Chosen One, Alan Turing. Turing showed that conceptually all computation can be expressed, simply and beautifully, in a state-based model [20].

Until the Plaid Brotherhood took up its holy quest to save the world, however, there was no realistic programming paradigm or programming language for performing holy, state-based programming, as Turing intended. We show how Turing's Machines can be implemented in a Hallowed subset of the typestate-based programming language Plaid. Our approach creates no garbage—instead, whenever objects are no longer needed in their current state, they are Saved and Born Again in a new holy state. Nor do we ever allow an object's field values to be destroyed by assignment; instead, new field values may be bestowed as part of the Sacrament of State Change.

The epistle herein, if followed and promulgated by those faithful to the One Plaiddish Way, can save the world!

ALASKA. THE WAY OF UNIVERSAL SALVATION

The One Pladdish Way of programming follows two simple rules. First, the wanton destruction of assignment is forbidden. All evolutions in the state of objects must be accomplished instead via the Sacrament of State Change. Second, objects may never be released as garbage; each object must be reborn in a new state when no longer needed.

ARIZONA. EXPERIENCING CLEAR

A programmer who has foresworn assignment and garbage, and who is following the One Plaiddish Way, reaches a empti-

[1]An idea published [1] in a conference so prestigious it has an exclamation mark in its name! please PLEASE give me tenure!

SIGBOVIK '10 1 April 2010, Pittsburgh, PA

ness of mind, spirit, and CPU that has been called "Clear" [14]. But is it possible to reach this happy state in Plaid practice?

St. Turing showed the way with his Universal Machines, which can encode any computation using the ideas of states. Through long study, secret handshakes, and lost symbols [2] we have developed a construction of Turing's Machines in the Plaid language, following the One Plaiddish Way.

```
state Cell {
    method getLeft() {
        left;
    }
    method getRight() {
        right;
    }
    val left;
    val right;

    method print() { ... }
}

state LeftEnd {
    method getLeft() {
        val me = this;
        val myLeft = new LeftEnd with Zero {
                                right = me;
                            };
        val myRight = this.getRight();

        this <- Cell { left = myLeft; right = myRight; };

        left;
    }
    // getRight(), etc. as in Cell
}
```

Listing 1: Modeling Tape Cells

Cells in a Turing tape are modeled as holy states which are connected to the cells on the left and on the right, and have additional operations such as print.

Of course, a Turing tape is infinite, which is difficult for finite minds and primitive programming models to effectively represent. While some among the unfaithful might suggest laziness to model an infinite data structure, Plaid theology holds that sloth is one of the Six Deadly Sins. We therefore apply the Sacrament of State Change to approach an understanding of the Infinite. A LeftEnd state is like a cell, but it has no cell to the left of it, yet. When we need the cell to the left, we create it as a new LeftEnd (in the initial, Zero state—see below), set its right field to the current object `me`, and transform the current object into an ordinary Cell (using Plaid's state transition operator, written `<-`). There are corresponding states for RightEnd and the Start cell of the tape (which is conceptually both a left and a right end).

```
state Zero {
    method writeZero() {}
    method writeOne() {
        this <- One;
    }
    method printVal() {
        java.lang.System.out.print("0");
    }
}
state One { ... // similar
```

Listing 2: Modeling Cell States

Each cell in a Turing tape can be in one of a fixed number of states. Here we consider two such states, Zero and One. Either a Zero or a One state is combined with the Cell state when a Cell object is constructed. If we are in the Zero state, writing a one with writeOne() transforms the current object into the One state. Each state also knows how to print an appropriate representation.

```
state Beaver2B {
    val cell;

    method update() {
        match (cell) {
            case Zero {
                cell.writeOne();
                val newCell = cell.getLeft();
                this <- Beaver2A { cell = newCell; };
            }
            case One {
                cell.writeOne();
                val newCell = cell.getRight();
                this <- Halt { cell = newCell; };
            }
        };
    }

    method run() {
        update();
        run();
    }
}
```

Listing 3: Modeling Machine States

A Turing machine is represented by one or more internal states, such as the Halt state or the Beaver2B state shown above (from the 2-state, 2-symbol "Busy Beaver" Turing machine). The machine's processing is represented by the run() method, which updates the machine's state and then continues running in the new state (which, of course, may have a different run() method). The update() method uses a match to find out if the cell at the machine's head is a Zero or a One. It then writes a value (one in this case), moves left or right, and transitions the machine into a new state (Beaver2A or Halt).

As the revelation above shows, any Turing machine can be expressed in the One Plaiddish Way. Since any program can be expressed as a Turing machine, we have proven beyond reasonable doubt that the One Plaiddish Way is a practical way to live one's programming life.

The key to salvation described above is available in the form of Plaid source code at the Plaid monastery web site[2].

ARKANSAS. ETERNAL DAMNATION

We note in passing that the creators of the Plaid language have not reached "clear" and have regrettably included the option of both assignment and functional programming in

[2] `http://www.plaid-lang.org/`

the language. The Supreme Revolutionary Plaid Council of Pittsburgh has declared a Fatwa against these features, and they shall not be used by the faithful. The penalty for violations is the eternal torment of programming in COBOL.

CALIFORNIA. RELATED WORK

We approve of Wadler's inclusion of an exclamation point in the title of a paper [21]. However, we feel that *changeing* the world is a rather modest goal; our ambition is instead to *save* the world!

The dangers of The Assignment are well documented [9]. As for the problem of garbage, recent work paints a grim picture of the world's future [19].

Saving the world has been a problem for a long time. It was notably tried over 2 millenia ago, and though the effort ended in the death of the protagonist, many believe the approach to have been successful [5]. Other, more recent (and highly misguided) approaches have, at the cost of great struggle [13], not only failed to save the world but may indeed have brought the end of the world [11] closer. The author notes evidence from *SIGBOVIK '09* reviews that that Nazis used SML [4], indicating that garbage was part of their nefarious plots.

Some attempts to save the world have been downright spellbinding [17], and have achieved fanstastic success at banishing evil. While most research has focused on saving our world, recent work considers ways to save other worlds as well [3].

Some believe the world can be saved if people would do only 50 simple things [12]. It is notable that #47 is to avoid garbage through recycling objects, as our State Change Sacrament accomplishes.

We save our discussion of research published in the most *distinguished* venues for last. MapReuse and MapRecycle [15] are very much in the spirit of our state-change based object reuse strategy. Finally, the One Plaiddish Way draws inspiration from the One True Coding Style [10], which also uses Holy languages, but which also blasphemously uses assignment in C++.

COLORADO. FUTURE WORK

In future work we hope to formalize the semantics of Holy States in Drunken Logic [18]. Should be fun!

CONNECTICUT. CONCLUSIONS

Believe not in the false prophets of functional and imperative programming. Holy states CAN save the world!

DELAWARE. ACKNOWLEDGEMENTS

The author gratefully acknowledges the other members of the Plaid Brotherhood for the development of the One Plaiddish Way and its support in the Plaid compiler. In addition, Miss Mouse provided moral support throughout the author's childhood, for which he is eternally grateful.

1. REFERENCES

[1] J. Aldrich, J. Sunshine, D. Saini, and Z. Sparks. Typestate-oriented programming. In *Onward!*, 2009.

[2] D. Brown. *The Lost Symbol*. Doubleday, 2009.

[3] J. Cameron. Avatar, 2009.

[4] J. Cette. Review of the one true coding style. In *The 8th Biarennial Workshop about Symposium on Robot Dance Party of Conference in Celebration of Harry Q. Bovik's 0x40th Birthday*, 2009.

[5] J. Christ. *The Holy Bible*. 33.

[6] A. Church. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58:354–363, 1936.

[7] Dictionary.com. Definition of execution. http://dictionary.reference.com/browse/execution.

[8] J. Donham. Compacting, composting garbage collection. In *The 6th Biarennial Workshop about Symposium on Robot Dance Party of Conference in Celebration of Harry Q. Bovik's 0x40th Birthday*, April 2007.

[9] C. Duguay. The assignment, 1997.

[10] J. M. (editor). The one true coding style. In *The 8th Biarennial Workshop about Symposium on Robot Dance Party of Conference in Celebration of Harry Q. Bovik's 0x40th Birthday*, 2009.

[11] Fluid. The end of the world. http://www.albinoblacksheep.com/flash/end.

[12] E. Group. *50 Simple Things You Can Do to Save the Earth*. Bathroom Readers Press, 1990.

[13] A. Hitler. *Mein Kampf*. Hurst and Blackett, London, 1939.

[14] L. R. Hubbard. *Dianetics The Modern Science Of Mental Health*. 1950.

[15] M. McGlohon. Mapreuse and maprecycle: Two more frameworks for eco-friendly data processing. In *The 8th Biarennial Workshop about Symposium on Robot Dance Party of Conference in Celebration of Harry Q. Bovik's 0x40th Birthday*, 2009.

[16] J. M. Newcomer and C. B. Weinstock. Cycle depletion—a worldwide crisis. In *The 6th Biarennial Workshop about Symposium on Robot Dance Party of Conference in Celebration of Harry Q. Bovik's 0x40th Birthday*, 2007.

[17] J. K. Rowling. *Harry Potter and the Deathly Hallows*. Bloomsbury, 2007.

[18] R. J. Simmons. A non-judgmental reconstruction of drunken logic. In *The 6th Biarennial Workshop about Symposium on Robot Dance Party of Conference in Celebration of Harry Q. Bovik's 0x40th Birthday*, 2007.

[19] A. Stanton. Wall-e, 2008.

[20] A. M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42):230–265, 1937.

[21] P. Wadler. Linear types can change the world! In *Programming Concepts and Methods*, 1990.