HMMMML2：超好意的に解釈するコンパイラ

中橋　雅弘[†]　宮下　芳明[†]

HMMMML2：Super-Generous Compiler

MASAHIRO NAKAHASHI[†] and HOMEI MIYASHITA[†]

本稿で提案するプログラミング言語 HMMMML2 における超好意的解釈コンパイラは，スペルミスを大幅に許容するとともに，複数の解釈が成り立つ場合はその全ての解釈を実行し，タブ切り替えによってユーザに選択させる．また，プログラムの正確さをスコア化し twitter でつぶやく機能も持つ．

1.　はじめに

安村はユーザ指向のプログラミング言語デザインの指針として Programming 2.0 を提唱し，これからのプログラミング言語は人間の特性をふまえて設計されなければならないと述べている[1]．この未来ビジョンは，特にプログラミング教育の観点からも重要な考え方であると筆者は捉えている．

しかし現在のほとんどのプログラミング言語は，極めて厳密な構文指向であり，守らなければならないルールが非常に多いだけでなく，それらをひとつでも間違うと実行結果を表示すらしてくれない．こうした制約はプログラミングを学び始めたばかりの人にとって大きな障害であり，学習に対するモチベーションの低下につながると考えている．本稿第二著者はこうした思想から「モチベーションを向上させる言語」としてこれまで HMMMML[2] を開発してきた．本稿では，この言語をさらに発展させた HMMMML2 ついて述べる．

2.　HMMMML

2.1　デザインポリシー

モチベーション向上を考える上で，まずプログラミングにおける学習意欲の低下要因を以下のように考えた．

プログラマにとって数十行のソースコードは「短い」ものだが，初心者にとっては，それでも長いものと捉えられている．行数だけでなく，その煩雑さもモチベーションを下げている要因と捉えた．たとえば一般に「3 回 Hello と書く」ためには，

int i; for (i = 0 ; i < 3; i++) { }

という書き方が一般的だが，「3 回繰り返す」という意味を表すには些か大げさな印象があり，可読性も悪く，書く側としてもやや面倒でミスを誘う表現である．

そもそも，初心者が書くプログラムはミスが多く，なかなか動かない．変数の宣言忘れ，インクルードし忘れ，スペルミス，セミコロン抜け，==と=の混同，大文字小文字，{ }の不整合の間違いは日常的である．

また，多様かつ高度な情報機器が身近に溢れている現代においては，指令したとおりに文字が表示されることからくる感動などはまずなく，“Hello World”の文字列から広大なプログラミングの「世界」を感じることは難しくなってきている．むしろ文字列の表示どころか，「円を描く」「jpg ファイルを表示する」「ドレミの音を出す」ことすら当たり前のように感じられてしまう．プログラミングに対するモチベーション向上を考えると，文字列表示を行う print 文による導入では不足である．

言うまでもなく，将来的には長いソースコードに対する耐性も身につけなければならないし，複雑なプログラミング表現もできるようにならなければならない．派手なものだけが善であるわけもない．プログラミングを極めるなら，いずれは汎用的な言語で実装を行う技術が必要になってくる．しかし，そういった鍛錬が持続的に推進されるためには，まずプログラミングの可能性や面白さを十分に体験しておく必要があると考えた．また，AT 車的な支援思想に則り，Squeak[3]や Scratch[4]，MAX/MSP/Jitter[5]に代表される直感性を追及したビジュアルなインタフェース，あるいは例示

† 明治大学　理工学部　情報科学科
Meiji University

プログラミング手法[6][7]までは検討せず，あくまでテキストを入力してコンピュータを動作させる範疇で支援のデザインを行うこととした．

これらの要件を満たすプログラミング言語として，HMMMML（Homei Miyashita's Motivating Multilingual Markup Language）をデザインした．エディタの「無限ループボタン」によってたった 1 行でも多様な表現が行える．「好意的解釈コンパイラ」は，セミコロン抜けや宣言忘れなどのミスも許容して実行される．｛｝を完全廃止し，ループや条件判定のためのタグ命令が用意される．C や Java ，Javascript など他言語と混在・混同して書いても実行できる．さらに，音声合成による文字列の読み上げ，インターネットを利用した検索や翻訳，MIDI 音の出力，Google マップの表示といった，「派手」な処理が一命令で実行可能である．

2.2 基本仕様

2.2.1 タグ命令

HMMMML で３回 Hello と表示するときには，「for タグ」を使用して以下のように書ける．

<for 3> print “Hello”</for>

カウンタとしての変数の宣言や初期値設定;条件式;増分処理を指定する必要がない．

2.2.2 無限ループによる１行プログラミング

マウスカーソル位置に半径 20 の円を描画するには

draw.circle(mouse.x, mouse.y, 20);

と書ける．エディタに備わっている無限ループボタンをオンにして実行すると，図 1 のようなペイントツールを１行で実現できる．ESC キーやウィンドウの閉じるボタンでループから脱出することができる．

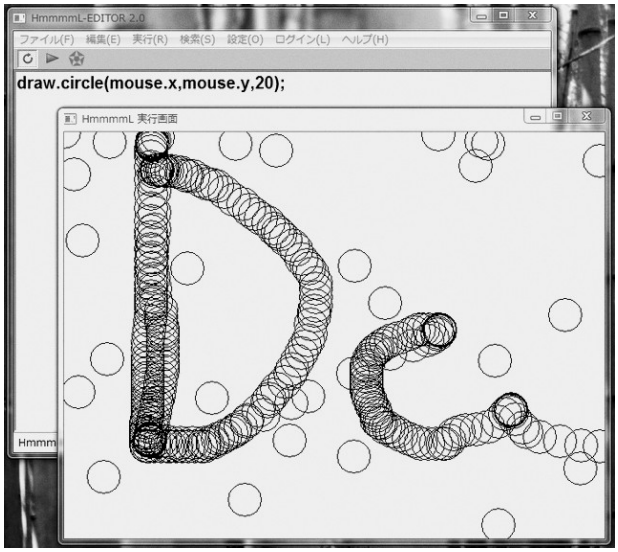

図 1 HMMMML を用いた１行プログラミングの例

2.2.3 好意的解釈コンパイラ

HMMMML はセミコロンの有無やドット抜け，括弧の閉じ忘れについても許容する「好意的解釈コンパイラ」を搭載しているため，先ほどのプログラムは

drawcircle(mousex,mousey,20

と書いても同じ結果を得ることができる．

if 文の条件式内で==の代わりに=を使用してしまっても==と解釈し，大文字小文字の混同や配列変数を含む宣言忘れについても許容して動作する．

2.2.4 多言語との混同

好意的解釈コンパイラは，for 文を C や Java の記法で書いたり，print の代わりに Javascript のように document.write(”); や Java のように System.out.println(“”);と書いても実行するなど，他の言語の様式と混在させたコードも解釈して実行することができる．

2.2.5 「派手」な命令群

say “Hello”と実行すると人工音声で“Hello"と発音し，google “Hello”と実行すると“Hello”の検索結果をブラウザに表示する．MIDI で音を鳴らすには play.note(ノート番号)，指定したサイズの Google マップを表示するには draw.map(横幅，縦幅，緯度，経度)と１行で行える．動画ファイル・音楽ファイルも全て play “ファイル名”で実行する．

3. HMMMML2

本章では，これまで推し進めてきた HMMMML をさらに拡張した HMMMML2 について説明する．

3.1 解釈箇所の実行後提示

「そこのゲッキョク駐車場を右に曲がって」と指示されたタクシー運転手は，どう対処するのがよいだろうか．「そんな駐車場はありません」と言って正しく言い直すまで右折しないのがこれまでのコンパイラ的思考であり，「月極駐車場のことを指しているんだな」と好意的に解釈して右折するのが HMMMML 的思考である．しかしこれでは，乗客は漢字の読み方を間違えたままであり，学習の促進を考えたとき，最も理想的なのは，とりあえず右折してから「ほんとはツキギメって読むんですよ」と小声で教えてくれる運転手ではないだろうか．

HMMMML は「好意的な解釈によってとりあえず実行する」ものだったのを，HMMMML2 では「好意的な解釈によってとりあえず実行し，その後でコンパイラがどこを修正したのかをユーザに教える」ものに発展させた．例えば if 文の条件式内で「a=1」と書いた場合，HMMMML2 ではこれを「a==1」と好意的に

解釈して実行した上で，ソースコードには「a=1 を a==1 に修正しました」とコメント文が入るようにしたのである（図 2）.

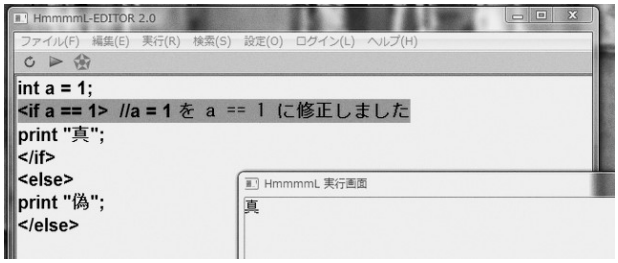

図 2 解釈結果の実行後提示

これまでのプログラミング言語やその開発環境では，エラーとなっているところに下線を表示し「実行前に直させる」インタラクションデザインとしていたが，HMMMML2 は「早く実行結果が見たい」というユーザの気持ちを優先させ，まず実行し，そのあとで正しい表現が何であったかを伝えるのである.

3.2 超好意的解釈コンパイラ

HMMMML2 では，さらに好意的な解釈を行う「超好意的解釈コンパイラ」を搭載した．ひとつのミスも許さない超頑固なコンパイラの対極の存在を生み出すことが出来れば，「超好意的」から「超頑固」までの広い範囲で支援度合の段階的調整ができるので，プログラマのスキル向上を促すインタラクションデザインにつなげられると考えた.

3.2.1 好意的な無視

はじめてプログラミングに挑戦する人に多いのが，全角スペースを入力してしまった場合である．これは見た目だけで気づくことは難しい．そこで HMMMML2 では，プログラムに直接関係ないと思われる全角スペースや２バイト文字は全てコメントとして好意的に無視するようにした．これにより，通常コメントを書く際に必要な//や/*～*/を忘れたとしても，それが２バイト文字であれば問題なく動作するようになった.

3.2.2 スペルミスの好意的解釈

HMMMML2 では，スペルミスに対しての許容幅を拡張した．予約語にない命令，例えば ptint “Hello”という文字列があったとき，これは print を意図して記述されている可能性が高いので，コンパイル時に print と置き換えて実行するようにした．コンパイラ内部のアルゴリズムとしては，予約語との一致率を計算して類似した予約語に置換している.

3.2.3 複数の解釈がなりたつ場合

HMMMML2 の言語仕様では，p で始まって文字列が続く命令が print しかないため，p “Hello”と記述しても print “Hello”と解釈される.

これに対し，s “Hello”と書いた場合は，say(人工音声で読み上げ）と show(ダイアログ表示)の２通りが考えられる．HMMMML2 では，複数通りの解釈がなりたつ場合は，その全ての解釈を実行することとした．それぞれタブを切り替えることで実行結果を見ることができ，「採用」ボタンを押すことでその解釈を選択することができる（図 3）.

図 3 複数解釈に対するタブ表示

たとえば図３において，タブ１を選択すると「Hello」が読み上げられ（say としての解釈）タブ２を選択すると「Hello」というダイアログが表示される（show としての解釈）．タブ１を選択して採用ボタンを押すと，ソースコードは

say “Hello” //s を say に修正しました

という内容に修正されている

3.2.4 解釈不能事例のアップロード

現状のシステムで好意的に解釈できないような特殊なエラーが発生した場合は，そのプログラムをサーバ上にアップロードする機能を備えた．これにより集められた多くのプログラムからエラーの典型を導き出し，超好意的解釈コンパイラの解釈機能強化に役立てることができる.

3.3 スコアリング・ランキング

HMMMML の発表時[2]において批判されたことのひとつは，好意的解釈機能の副作用として，ユーザがこれに慣れてしまい，正確に書かなくなってしまう恐れがあることであった．HMMMML2 は，これを避けるためにプログラムの正確さの評価を行い，スコアとして換算し twitter でつぶやくようにした.

将来的にはスコアのランキングを表示できる機能を設ける．ランキングはユーザ個人のものだけでなく，全国のランキングを閲覧できるようにする．これによ

ってユーザ同士が健闘を讃え合うことができる．また，完璧なプログラムに近ければ勲章やトロフィーを表示するといった機構も用意する．これによりユーザがモチベーションを低下させることなく，さらにより良いプログラムを書こうという意欲を出せるよう工夫したい．現状ではスコアは命令数に比例，修正数に反比例する換算式を用いている．

4. 展望

前述の通り，超好意的な解釈を行うコンパイルまで作ってしまえば，段階的に解釈を厳しくしていくのは容易である．このため，今後は超好意的解釈-超頑固解釈を調整できるスライダを実装する予定である．そして，好意的解釈に依存しないほど高得点を得ることが出来るシステムにすれば，支援のための補助輪をユーザが自ら外し，より高度なプログラミングの世界に飛び出してくれるはずである．

2 章で述べているように，本システムは筆者らが主観的に定めたデザインポリシーに基づいて試作されたものにすぎない．こうしたプロトタイプの段階から多くのフィードバックを与えてくれたプログラミングシンポジウムに感謝するとともに，次稿からは試作報告にとどまることなく，これまで数多く提案されてきているエンドユーザプログラミング支援手法に言及した議論を行っていく予定である．

参 考 文 献

1) 安村通晃．Programming2.0：ユーザ指向のプログラミング，情報処理学会夏のシンポジウム 2006.
2) 宮下芳明．プログラミングに対するモチベーションを向上させる新言語 HMMMML の開発，第 51 回プログラミング・シンポジウム予稿集，pp.57-64，2010.
3) Cardelli, L. and Pike, R. Squeak: a language for communicating with mice. In Proceedings of the 12th Annual Conference on Computer Graphics and interactive Techniques SIGGRAPH '85. 1985.
4) Scratch: http://scratch.mit.edu/
5) Max/MSP/Jitter http://cycling74.com/products/maxmspjitter/
6) 増井俊之. インターフェイスの街角(47) - 例示プログラミング. Unix Magazhine, Vol. 16, No. 11, 2001.
7) Hartmann, B., Wu, L., Collins, K., and Klemmer, S. R. Programming by a sample: rapidly creating web applications with d.mix. In Proceedings of UIST2007. 2007.

質 疑 応 答

Q 川中：解釈の幅が２のｎ乗通りあるような場合，ユーザはどうしたらよいのでしょうか？

A 中橋：そのように解釈の幅が広がってしまった場合，有力な候補を探し解釈の数に制限を設けて，解釈するのを止めてあげなければならないと考えています.

Q 中山：このスコアリングの評価手法ではスコアが高かったからといって，必ずしもプログラムの正しさに直結しないと思うのですが，アルゴリズムの正しさの評価などは何か考えていますか？

A 中橋：アルゴリズムの正しさの評価は最終的に人が判断しないとわからないと考えていて，やはり文法などからしか評価はできないと考えています.

Q 伊知地：間違いを許すのだけれど，それを正す時に本当はこう書くのだぞと警告を表示するなどといったことは考えていないのでしょうか？

A 中橋：提示の仕方を増やすのはおもしろいと思います．例えば指導教官モードではそのように警告で指導されたり，またツンデレモードなども用意するとおもしろいと思います.

Q 岩崎：一回目の間違いは超好意的に解釈するが，それが何回も続いた場合は教育的指導をするといったように変化を持たせるとおもしろいのではないかと思いました.

A 中橋：エディタがユーザ個人に対応していくというのはとても面白いと思います．使いこんでいくうちにその人なりの間違い方などが解釈できる仕組みを考えたいです.

Q 伊集院：好意的度合いのスライダを数値化したとして，99，98 といったケアレスミスだけを直してくれる数値にほとんどの人が設定するような世界になりはしないでしょうか？

A 中橋：まだ検証していないのでわからないのですが，そのようにある一定の値に集まるのかというのにもとても興味があります.

Q 脇田：間違えたらその場で勝手に構文を拡張していくというのは考えないのですか？

A 中橋：プログラムを書いているという実感は持たせたいと考えているので，構文を勝手に拡張するまではさせないです.

Q 副田：解釈が間違っていた場合はどうするのですか？

A 中橋：現時点では解釈前のファイルを開くしかないです．これに関してはさらに対策が必要だと思います.