4A-01

関数型言語 SML#による開発支援のための Eclipse プラグインの開発

高梨　勝敏　　菊地　大介　　齋藤　邦夫　　手塚　大

(株)日立ソリューションズ東日本

1. はじめに

SML#は関数型言語の持つ生産性・信頼性の高さに加えて，C 言語外部関数インタフェース，言語組み込み SQL などの高機能性を持つ [1]．プログラミング言語の普及促進のためには，言語の特長の認知に加えて，開発環境の充実が重要である．本論文では開発環境に焦点を当て，SML#でのプログラム開発を支援する機能の開発と評価を行う．

2. 開発支援項目

開発環境の利用によりプログラミング言語の理解が向上するという指摘 [2]がある一方，複雑化しすぎると使いこなすために学習等のコストがかかるという指摘もある [3]．そこで SML#による開発を支援する際に必要になる機能を整理した．その機能を表 1 に示す．詳細設計でのモジュールインタフェース設計以降，コーディング，テストの各工程で必要となる基本機能と，そこで SML#の特長である型推論機構を有効活用するための機能を挙げた．

3. Eclipse による SML#統合開発環境

利用者数が多く，プラグインにより機能が拡張できる Eclipse[4]を使用して，SML#の統合開発環境（以下 IDE）を開発することとした．

3.1. Eclipse プラグイン

Eclipse で提供されているプラグイン SDK を用いてプラグインを開発した．SML#コンパイラは対話モードでの情報提供が充実しており，ユーザは SML#の式を評価させて値や型の情報を得ることができる．そこで，プラグインはユーザに代わって SML#コンパイラの対話モードとデータを授受する．コンパイラによる評価結果を加工して IDE の画面に表示する．

(1) コード補完

ユーザがコードを途中まで入力すると，その後の入力候補が表示され，コード入力を支援する機能である．ユーザがモジュール名とピリオドを入力し，関数名を入力しようとしたタイミングがコード補完機能の開始点となる．処理手順を図 1 に示す．IDE は当該モジュールを変数に束縛するコードを生成する．変数束縛は関数型以外のプログラミング言語では代入に対応する．例えば，モジュール Math では structure a = Math を生成する．当該コードをコンパイラの対話モードに入力して評価させると，Math モジュールの持つ関数と引数の型の一覧が得られる．IDE は，当該評価結果を関数呼び出しコードのリストに加工し，コード補完の候補としてユーザに提示する．コード補完の画面例を図 2 に示す．

表 1　開発支援に必要な機能

	機能	SML#に特化した機能
1	プロジェクト単位でのソース管理	-
2	ソース色分け表示	-
3	自動インデント	-
4	コードの雛形挿入	-
5	コード補完	モジュールの評価結果でコードを補完
6	定義行への移動	-
7	関数や変数の情報表示	-
8	コンパイラのコマンド実行	-
9	デバッグ	ソースを評価し，仮想的にステップ実行
10	ドキュメント生成	-

(2) デバッグ

SML#にはデバッグ機能が用意されていない．そこで対話モードを用いて簡易的にステップ実行や変数の値の確認などができるようにした．ユーザがコーディングしたソースに対し，デバッグメニューを選択するとデバッグ機能が開始する．処理手順を図 3 に示す．SML#のプログラムソースを式（関数，変数の束縛）毎にコンパイラに評価させる．例えば，関数 myfunc がコーディングされている場合，fun myfunc=以下の内容を評価させる．変数の束縛は，val a=以下の束縛内容を評価させる．評価結果として，変数の値と型が得られる．IDE では，ユーザのステップ実行操作毎に，変数の値と型を順次表示することにより，仮想的なステップ実行を可能とした．

Eclipse plug-in for functional programming language SML# developers

TAKANASHI Katsutoshi, KIKUCHI Daisuke, SAITOU Kunio, TEZUKA Masaru

Hitachi Solutions East Japan, Ltd.

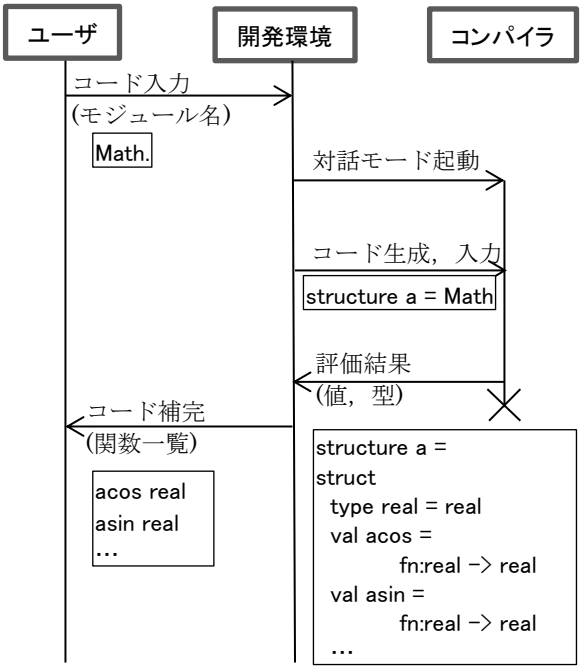

図 1　コード補完での対話モードとの通信

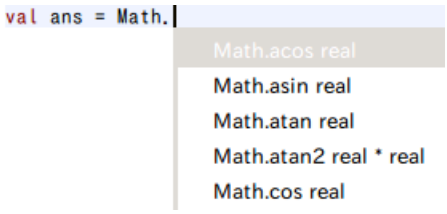

図 2　コード補完の画面例

4.　機能の評価

試作した IDE を，SML#のプログラム部品の設計・開発に適用し，機能を定性的に評価した．

コーディング時はコード補完機能によりモジュールの評価結果を元に候補を表示することで，候補提示がユーザにとって分かりやすくなり，コーディングの効率が向上した．

デバッグ時，簡易ステップ実行によりデバッグ効率が向上した．コンパイラは関数内の各ステップの評価結果を出力しないため，関数内のステップ実行ができない．これらの機能の実現には SML#コンパイラの改造が必要となる．

本機能の評価はまだ十分ではないが，今後，より多くの開発で利用しながら評価，改良を進める．

5.　おわりに

SML#の開発環境を Eclipse のプラグインとして試作した．設計・開発の基本機能を満たし，SML#の特長である型システムによる型推論を活用したコーディング，デバッグおよびレビュー

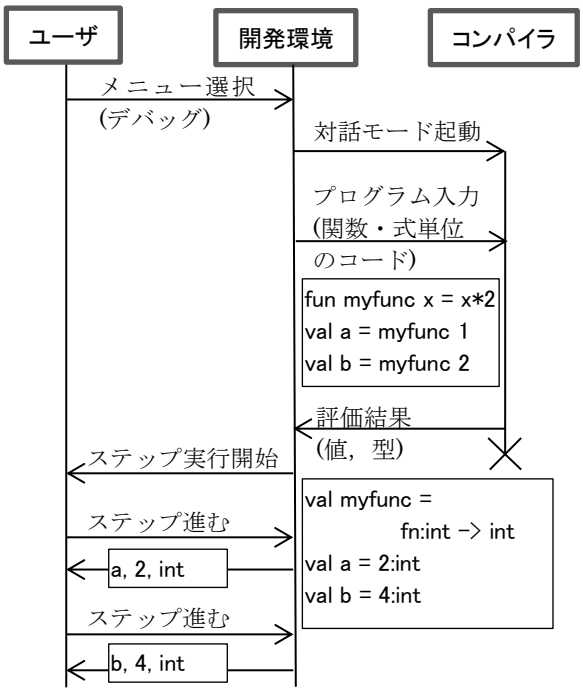

図 3　デバッグでの対話モードとの通信

支援を実装した．設計・開発で試用し，機能の実現性を確認したが，デバッグ時のステップ実行に制限がある．今後は機能の改善のほか，開発プロジェクトでの利用を通じて品質・生産性向上の定量評価を行っていく．

6.　謝辞

本研究は文科省の委託事業「高機能高可用性情報ストレージ基盤技術の開発」の中で実施している．

参考文献

1) Atsushi Ohori, Katsuhiro Ueno, Kazunori Hoshi, Shinji Nozaki, Takashi Sato, Tasuku Makabe, Yuki Ito, SML# in Industry: A Practical ERP System Development, Proc. ACM ICFP Conference (2014)

2) Davi Felipe Russi, Andrea Schwertner Charao, Integrated Development Environments as a Supporting Tool for Teaching the Haskell Programming Language, RENOTE - Revista Novas Tecnologias na Educação, Vol.9, No.2 (2011)

3) Iyad Zayour, Hassan Hajjdiab, How Much Integrated Development Environments (IDEs) Improve Productivity?, Journal of Software, Vol.8, No 10 (2013), pp.2425—2431

4)Eclipse, https://www.eclipse.org/home/index.php , Accessed 2015 年 11 月 11 日