Mini Konoha: 最小構成から目指すディペンダブルスクリプト言語の設計

井出真広[†1]　菅谷みどり[†2]　倉光君郎[†2]

近年、スクリプト言語の適応分野は広がっており、適応分野ごとに設定された問題の解決にスクリプト言語が利用されている。多くの開発者はプログラミング言語の文法に対して小さな不満を持っており、今もなお、独自のドメインにあわせて DLS の新規開発を続けている。本発表で提案する Sugar パーサはスクリプトによってユーザ定義の構文を拡張する柔軟な手段を提供する。本発表では Sugar パーサをスクリプト言語に統合した MiniKonoha 言語を紹介する。

1. はじめに

スクリプト言語は、プログラミングしやすさや生産性に注目が集まり、広く採用が行われてきた。そして今日ではスクリプト言語の適応分野は Web アプリケーション開発からゲーム開発分野、システムコンフィギュレーションなど多様なドメインについて利用され、産業界からのスクリプト言語への関心は高まっている。

我々は、静的型付けの特徴をもつ KonohaScript[1]のオープンソース開発を行ってきた。現在、JST/DEOS プロジェクトにおいて、障害対策のスクリプト化、プロセス科を研究してきた。KonohaScript は、静的型付けによるスクリプトの型検査機能を活かして、より高信頼スクリプト実行基盤である D-Script エンジン[2]として開発を進めている。

ディペンダビリティ分野では、一般に、より高いソフトウェア信頼性を実現するため、複雑な仕様や実装は忌避される傾向にある。これは、スクリプト言語仕様や処理系においても当てはまる。2012 年 3 月に開催された DEOS 国際シンポジウムにおいても、D-Script への要望として、ユーザの書き間違えや処理系のバグを最小化するため、コンパクトな言語が強く求められた。

一方、D-Script が対象とするコンピュータシステムの障害対策は、多様な要望がある。汎例をあげると以下のとおりである。

- （組み込みシステムを含め）リソースの負荷の少ないモニターが書けること
- 大量のログを処理して障害原因の解析が行えること
- 分散するシステムを連動して制御できるようにすること
- 実行時のふるまいの変更を保証するセキュリティ機構
- 複雑な処理をルールとして簡単に書けること

障害対策スクリプトは、多様なドメインで利用されるスクリプト技術であり、ひとつの言語ですべての解決をみることは難しい。現在、Bourne Shell や Perl, Python, Ruby など、様々なスクリプト技術が併用されているのは、特性にあわせた使い分けによる。また、ポリシー記述やコード生成の手法も広く使われている。これらの要望に対し、様々なドメイン問題に対する高信頼なスクリプト実行基盤をどう作るべきなのか、我々の探求すべき課題となっている。

2. Mini Konoha +文法拡張可能なパーサ

我々は、最小言語構成からはじめ、拡張することでさまざまなドメインに容易に適合させやすいスクリプト言語の設計・実装を目指している。拡張手法は、SugarJ が実現したライブラリによる文法拡張を採用し、文法拡張をライブラリとして記述することを目指している。たとえば、クロージャの文法を使いたければ、次のように Sugar ライブラリとしてインポートすることができる。

```
import sugar.closure.*;
```

Sugar パーサは、文法拡張可能なパーサの呼称である。ユーザに対し、トークナイザ、抽象構文木を構築するための API と型検査器へのインタフェースを提供する。図 1 は、Sugar パーサの概要である。ユーザは、トークンに対するパターンマッチ、トークン列に対するパターンマッチとステートメントへの変換方法をスクリプトで記述することができる。

- トークナイザ（任意の文字に対する）
- ステートメント（任意のパターンに対する）
- 式（単項演算式、二項演算式）
- 型検査器

Mini Konoha は、Sugar パーサ付属のスクリプト言語として提供される。同時に、Mini Konoha 自体のパースは、Sugar パーサで行うことにより、Sugar パーサを通して、Mini Konoha を Mini Konoha で拡張することが可能になる。さらに、Sugar パーサが提供する抽象構文木へのインターフェースは、コード生成のスクリプト化も可能にし、様々なターゲットシステム上で実行可能なコードが生成可能とな

†1 横浜国立大学
†2 横浜国立大学 / 現在 日本科学技術振興機構 CREST

る。現在の実装では MiniKonoha のインタプリタ(MiniVM)に加え、LLVM を用いたネイティブコード生成(JIT コンパイラ)、JVM, CLI, JavaScript へ変換可能となっている。

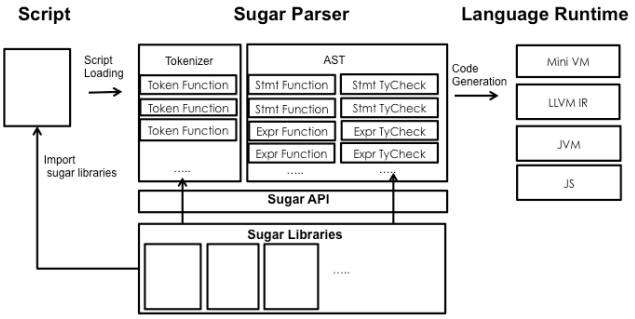

図 1 MiniKonoha と Sugar Parser

3. 最小言語仕様と安全性の議論

Mini Konoha は、KonohaScript をベースとして、言語仕様を最小化している。ただし、Konoha の特性を継承し、静的型付けとして、Python や Ruby など、動的型付け言語の機能は静的型付けの上で dynamic 型などでサポートする。

サポートしたもの
- 型宣言
- Object, Boolean, Int, String, Array[T], System, ..
- if-else文, return 文
- 関数(メソッド)定義
- 定数定義
- 関数(メソッド)コール
- (Cスタイルの)関数オブジェクト
- 演算子: + - * / % < > == !=
- /* コメント */ // コメント
- @アノテーション

サポートしなかったもの
- サブタイプ, 型推論
- Float, {Input/Output}Stream, Iterator[T]
- 代入
- while文 (do,while文)
- class文
- 例外処理 (try/catch)
- グローバル変数
- new 演算子, 配列 [] 演算子
- クロージャ(function 演算子)
- '' シングルクオートのリテラル
- // 正規表現リテラル

図 2 Mini Konoha vs. KonohaScript

図 2 は、Mini Konoha が KonohaScript に比べて、サポートしたものサポートしなかったものの比較である。原則、Mini Konoha は、Sugar Parser によって、文法拡張可能な十分な機能があれば十分である。たとえば、while 文が拡張することができれば、while 文を拡張したのち、while 文を用いた文法拡張を行うことができる。ただし、Mini Konoha の言語仕様として、プログラミング言語としての書きやすさも考慮にいれる必要がある。

我々は、最小セットの機能を選ぶための判断基準は、その機能が安全かどうかである。たとえば、while 文は、ユーザが容易に無限ループを書くことができ、無限ループのプログラム実行前の静的検証は難しいことが知られている。そのため、while 文を制限すれば無限ループは回避できる。Mini Konoha は、文法をライブラリとしてコントロールできるため、このような対応が可能になる。

表 1 言語文法と安全性

文法	論点	安全性
if 文	問題なし	○
while 文	無限ループとなる可能性あり	×
代入	副作用の可能性あり	?
グローバル変数	メンテナンス性低下、Race Condition 発生	×
new 演算子	大量メモリ消費 (GC 発生頻度)	×
配列 []	領域を超えたアクセスの危険性あり	×
/ 演算子	0除算のリスクあり	△

我々は、言語文法の安全性を保証ケースを用いて議論し、Mini Konoha 言語処理系とその上で動作するスクリプトの安全性の検証する作業を進めている。

4. おわりに

本論文では、まずスクリプト言語を構成するための最小の文法設計を行い、また拡張な形で言語ランタイムを設計することで、プログラミングのさまざまなドメインに容易に適合させることができる言語を設計・実装した。今後は、細分化された文法ライブラリと安全性評価を行い、ディペンダブルなスクリプト言語設計を目指して行く予定である。

謝辞 本研究は,JST/CREST「実用化を目指した組込みシステム用ディペンダブル・オペレーティングシステム」領域の研究助成を受けて行われた.

参考文献

1) Kimio Kuramitsu: KonohaScript: static scripting for practical use. OOPSLA Companion 2011: pp. 27-28, 2011.
2) 倉光君郎. 拡張性のある組み込みアプリケーションを実現するスクリプティング言語の開発, 情報処理学会論文誌. 51(12), pp. 2185-2194, 2010
3) Sebastian Erdweg, Tillmann Rendel, Christian Kastner, and Klaus Ostermann. SugarJ: Library-based syntactic language extensibility. In Proc of SPLASH2011, 2011.