プログラミング言語独立なプログラム解析基盤

桑　原　寛　明[†1]

本稿では，プログラミング言語に柔軟にプログラム解析技術を実現する仕組みの IDE における有用性を議論する．対象言語独立なプログラム解析ツールの開発を支援する SchemaAE を紹介し，コーディング検査器における応用例を示す．

A Language-Independent Program Analysis Platform

Hiroaki Kuwabara[†1]

In this paper, we introduce SchemaAE which supports to implement language-independent program analysis, show an application of SchemaAE to coding checkers, and discuss advantages of language-independent program analysis on IDEs.

1. はじめに

最近のソフトウェア開発では Eclipse などの統合開発環境 (IDE) の利用が一般的になっている．多くの IDE は複数のプログラミング言語に対応しており，Web アプリケーションのように複数のプログラミング言語を利用するソフトウェアの開発も同一の IDE 上で行うことができる．Eclipse であれば，Java 言語向けの JDT，C 言語向けの CDT，JavaScript 言語向けの JSDT などが提供されている．

IDE は，構文チェック，コード補完，リファクタリングなどの機能を実現するために，内部に言語モデル（基本的には抽象構文木と記号表）を持っている．加えて，エディタなどのユーザインタフェースを構成するウィジェット群も整備されている．そのため，IDE は新しいプログラム解析技術を実現するツールの開発基盤としても利用されるようになっており，IDE をツールプラットフォームとみなすこともできる．

IDE が対応する複数のプログラミング言語それぞれを対象に同じ解析技術を実現するには，通常はそれぞれの言語ごとに解析技術を実装する必要がある．IDE が複数の言語に対応していることは，IDE が言語の区別なくプログラムを操作する仕組みを備えていることを意味するわけでない．これは **Sapid** のような CASE ツール・プラットフォームでも同じである．

本稿では，具体的なプログラミング言語から独立した形で解析技術を実装するための仕組みとツールプラットフォームにおける有用性を議論する．プログラム解析を行うツールを言語モデルから独立して実現するためのフレームワークである SchemaAE[3] を紹介し，コーディング検査器における応用を示す．言語独立なプログラム解析基盤の可能性や，言語独立に実現できる具体的な解析技術について議論したい．

2. SchemaAE

独自の言語モデルを持つツールプラットフォームを利用したプログラム解析技術の実装では，必要な情報を言語モデルから抽出して解析を行い，解析結果を表示する．この時，特定の言語モデルに依存する形で情報の抽出から解析まで行うのがもっとも単純だが，同じ解析技術を他の言語モデルを対象に実現する際には再実装と同様な作業を行うことになる．複数の言語を対象にすることを初めから計画しているのであれば，**図 1** のように解析に必要な情報を具体的な言語モデルに依存しない抽象モデルとして定義して解析技術を実現し，各言語モデルに対して抽象モデルと対応付けるアダプタを作成することが考えられる．この方法であれば，図 1 の点線で囲まれたアダプタだけが言語モデルに依存し解析技術の実装は依存しない．しかし，アダプタを各言語モデルごとに作成しなければならず，この作業は機械的な作業も多く楽しいものでもない．

この問題に対して，言語モデルに柔軟にプログラム解析技術を実装するためのフレームワークである

†1 立命館大学情報理工学部
College of Information Science and Engineering, Ritsumeikan University

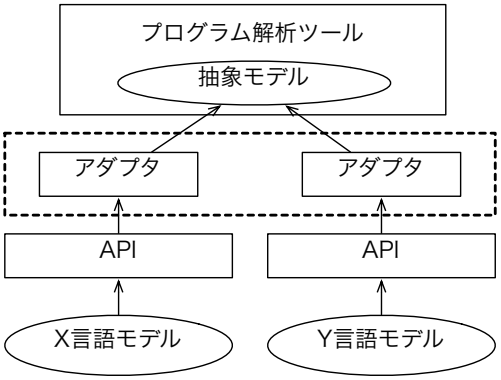

図 1 複数言語に対応するツールの構成

SchemaAE が提案されている[3]．SchemaAE の中核は，抽象モデルの定義，言語モデルの定義，および抽象モデルと言語モデルのマッピング情報の 3 つから，抽象モデルに従って言語モデルにアクセスするためのオブジェクト群を生成する生成系である．SchemaAE を利用すれば図 1 のアダプタはマッピング情報から自動的に生成される．SchemaAE の応用として，関数クロスリファレンサ[3] や GUI コードを異なる GUI ツールキットへ移植する手法[2] が提案されている．

3. コーディング検査器における応用

我々の研究グループでは C 言語向けあるいは JavaScript 言語向けの拡張可能なコーディング検査器を提案している[1),4)]．また，検査対象に依存しないコーディング検査器基盤についても研究開発を進めている．これらのコーディング検査器では，検査ルールの入れ替えや，新しい検査ルールの作成が可能である．

ある言語向けの検査ルールを作成していると，他の言語に対しても同じ検査が有効であることに気付くことがある．例えば，空白にタブ文字が使用されていないことの検査や，C 言語や Java 言語，JavaScript 言語などプログラムの見た目が C 言語ライクな言語における代入演算子の前後に空白が挿入されていることの検査などである．これらの検査のように検査対象の言語仕様にそれほど依存しない検査は言語独立に実現できると便利である．

空白にタブ文字が使用されていないことを検査する場合，ソースコードを構成する空白のみに着目すればよい．そこで，以下のような抽象モデル[⋆1]を利用して検査アルゴリズムを実装する．

```
<!ELEMENT Source (sp*)>
<!ELEMENT sp     (#PCDATA)>
```

sp はひとかたまりの空白を表す．Source はソースコードを表し，ソースコードは複数の空白から構成される．検査では，すべての sp をトラバースしながらタブ文字を含むか調べる．**Sapid** の JSX-model を利用して JavaScript プログラムを検査するのであれば，JSX-model の File と sp をそれぞれ抽象モデルの Source と sp にマッピングし，SchemaAE を用いてアダプタを生成すればよい．JX-model を利用して Java プログラムを検査する場合や，CX-model を利用して C プログラムを検査する場合も同様である．

4. おわりに

IDE の多くは複数のプログラミング言語に対応しているが，同じプログラム解析技術を個々の言語ごとに実装することは手間である．解析技術を言語独立に実現する仕組みがあれば，開発の手間が削減でき，解析対象の言語を追加することも可能になるため有用である．本稿では，コーディング検査器における例を示した．しかし，言語から独立して実現可能な解析技術はどのような技術なのか，そのような解析技術がどの程度存在するのか，明らかにする必要がある．

抽象モデル上のトラバースはマッピングされた具体的な言語モデルの API を利用して実現される．そのため SchemaAE では，要素の取得，ある要素に関連付けられた要素の取得，属性の取得，要素の文字列表現と位置情報の取得，などの機能を言語モデルの API に要請する．既存の API がこの要請を満たさない場合の対応についても考える必要がある．

参 考 文 献

1) 桑原寛明，末次　亮，山本晋一郎，阿草清滋：拡張可能な JavaScript 向けコーディング検査器，ソフトウェア科学会第 27 回大会，6C-1 (2010).
2) 水野佑基，金子伸幸，中元秀明，小川義明，山本晋一郎，阿草清滋：GUI 抽象化規則を用いたモデル生成手法，情処研報 (SE)，2006(35), pp. 121–128 (2006).
3) 新美健一，山本晋一郎，阿草清滋：抽象ソフトウェアエレメントによる CASE ツール開発のためのフレームワーク，信学技報 (SS)， Vol.103, No.582, pp.19–24 (2004).
4) 大須賀俊憲，小林隆志，間瀬順一，渥美紀寿，山本晋一郎，鈴村延保，阿草清滋：CX-Checker: C 言語プログラムのためのカスタマイズ可能なコーディングチェッカ，ソフトウェアエンジニアリング最前線 2009，近代科学社，pp.119–126 (2009).

⋆1 ここでは読みやすさを優先して DTD を利用した