B-008

C ソースコード静的解析のための問い合わせ言語 CXmlPyQuery の改良と評価
Improvement of query language CXmlPyQuery for static analysis of C source code and its evaluation

河合 勇太朗[1)] 福原 和哉[2)] 猪股 俊光[1)] 杉野 栄二[1)] 今井 信太郎[1)]
Yutaro Kawai Kazuya Fukuhara Inomata Toshimitsu Sugino Eizi Imai Shintaro

新井 義和[1)] 成田 匡輝[1)]
Yoshikazu Arai Masaki Narita

1 はじめに

ソフトウェア開発では，製品出荷前のソフトウェア検査が品質向上の観点から非常に重要であり，筆者らは ANSI-C で記述されたソフトウェアを対象とした検査ツールのためのフレームワークを提案している．そこではソースコードは CXml（中間表現としての XML 形式）[1][2] で表され，CXml のもとで静的解析手法が適用される．そのための問い合わせ言語として，CXmlPyQuery（API や処理系からなる）を開発した [3].

本研究では，CXmlPyQuery の改良と評価を試みた．

2 中間表現 CXml モデルと CXmlPyQueryTool

2.1 解析対象とする項目

開発した CXml と CXmlPyQuery を用いると，たとえば，コードレビュー時のコーティング規約に準拠しているかどうかなどの検査が可能である．本研究では，コーディング規約として，CERT-C セキュアコーディングスタンダード [4]（以下, CERT-C）を対象とする．

2.2 CXml モデル

CXml は, CXmlDTD で定義され，C ソースコードの曖昧さを排除して記述するための XML 形式の記述言語である. CXml は XML 文書の特性を備え，可読性，可搬性に優れ，各種 XML ライブラリによる検索も可能である．

2.3 CXmlPyQueryTool

CXmlPyQueryTool は，図 1 のように，CXml で記述された C ソースコードと，CXmlPyQuery によって記述された問い合わせ記述が入力されると，IronPython[5] と LibCXmlQuery などで解析し，その結果を出力する．

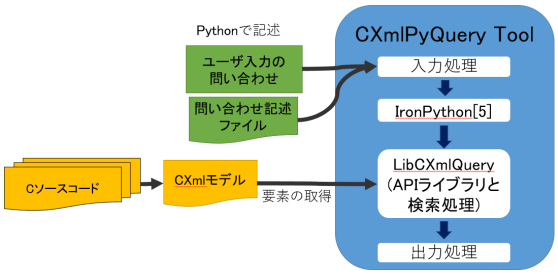

図 1 CXmlPyQueryTool 概要図

2.4 CXmlPyQuery の API

問い合せ記述で使用される CXmlPyQuery の検索用 API の機能は，次のように基本的 API と合成的 API に大別される．ここで，( ) 内は API の個数である．

- 基本的 API（計 66）
  - 構文要素の取得 ∗（49）
  - 現在要素の家族ノードの取得 (9)
  - 文字データの取得 (1)
  - 要素と属性の表示 (1)
  - 属性値の取得 ∗ (1)
  - 定数値の進数変換 (3)
  - 要素数の表示 (1)
  - Xpath での要素集合取得 (1)
- 合成的 API（計 3）
  - 指定パターン要素の取得 (1)
  - 直前・直後要素の取得 (1)
  - 関数の子要素の取得 (1)

これらの中には使用頻度が低い API，限定的な場合でしか使用されない API，さらには，XPath を利用している API もあり，本研究ではこれらの API（上記 ∗ 印の付いている機能の一部の API）の改良を図ることにした．

3 CXmlPyQuery の改良

3.1 API の使用頻度

CXmlPyQuery がもつ API を改良するにあたって，CERT-C の検査項目を記述したときの API の使用頻度や記述性を次の作業を通じて確かめた．ここで，対象とした検査項目は，既に CXmlPyQuery で記述可能であるとされたものである [3].

1. 合成的 API で記述されている箇所を基本的 API で書き直す.
2. Xpath を利用している箇所を基本的 API で書き直す.
3. API の使用頻度を調べる.

その結果，基本的 API の中でも，C 言語用の API は，1 つの検査項目での使用回数が少なく，XML 用の API は使用回数が多いことわかった．ここで，C 言語用の API は，C 言語の構文要素に依存して検索するためのもの，XML 用の API は XML の木構造に依存して検索するためのものである．

3.2 基本用 API の見直し

C 言語用の基本的 API で使用頻度の少ないものの中で，次の条件を満たすものに着目し，それらを合成して新たな API とした．

- 使用回数が 3 回以下
- 1 つの検査項目での使用回数が 1 回
- 特定要素の式を取得する API は除く

最後の項目は，検索すべき要素が，木構造のもとで，あるノードの子や子孫などの中に含まれることが多いからである．新たな API とするにあたっては，対象となる API を一括するのではなく，「宣言，メンバ定義，特定

1) 岩手県立大学大学院ソフトウェア情報学研究科
2) 岩手県立大学共同研究員

要素」に分けて合成することとした．その結果，新しくもうけた API を表 1 に示す. 表の各行の下段は合成された API である．

表 1　新たな API–その 1–

宣 言	node_unidecl_syntax()
	node_decl_typedef(),node_decl_union(),
	node_decl_enum(),node_decl_structure(),
	node_decl_empty()
メンバ定義	node_unidecl_member()
	node_decl_eum_member(),
	node_decl_member()
特定要素	node_unified()
	node_goto(), node_empty

一方，XML 用の基本的 API については，次の条件を満たすものから新たな API を作ることとした．

- 使用回数が 10 回以上
- 同じ引数が与えられて使用される回数が 10 回以上

これらの条件にあてはまるのは，属性取得のための Attribute() であり，これを表 2 のように，引数として与えられることが多い「型」と「参照先」のための API に分けることにした．

表 2　新たな API–その 2–

型取得	attr_type()
参照先取得	reference_node()

このうち，reference_node() の引数は参照先の要素の検索範囲の指定が可能である．

4　評 価

4.1　検査項目の拡大

CERT-C に含まれている検査項目のうち，改良前の API を用いることで，18 個が検査可能であった [3]．今回，改良した API を使用したことで，新たに 5 個が検査可能となった．これは次の理由による．

ある検査項目を調べるには，ソースコードの中に検査項目に該当する a) 構文要素やコードパターンを抽出するための問い合わせ記述と，b) 抽出されたコードが（検査項目によって指定された）条件を満たすかどうかを判定するための問い合わせ記述とが必要になる．改良前の API では，a) の問い合わせ記述による出力が，抽出された構文要素やコードパターンに関する情報（CXml モデルの要素としての ID や属性値など）が限られること，b) において自動的な条件判定をするのが難しい様式で出力されることから，１つの問い合わせ記述では対応できない検査項目が存在していた．それが，今回の API の改良により，a) によって抽出された情報を，b) の条件判定で利用できるようになり，検査可能な項目が増えた．

今回，検査可能となったのは，DCL30, EXP35, ARR33, ARR37, FLP36 の 5 項目である．このうち，DCL30 についての問い合わせ記述を次項で述べる．

4.2　記述例：CERT-C DCL30

CERT-C の DCL30「適切な記憶域期間でオブジェクトを宣言する」を検査するための問い合わせ記述（一部分）を以下に示す（一部折り返し）．

```
doc = load()
func = doc.node_def_function()
exp = func.node_exp_assign()
right=exp.child("右辺式").
      node_exp_variable().child()[0]
ref_right = right.reference_node(func)
isAddressAssign = False
if str(exp.ancestor("関数定義").attr("id"))==
 str(ref_right.ancestor("関数定義").attr("id")):
  if ref_right.descendant("配列型"):
   if IsArrayAddressAssign(ref_right,
      exp.child("右辺式")):
    isAddressAssign = True
  elif ref_right.descendant("ポインタ型"):
   if IsPointerAddressAssign(ref_right,
      exp.child("右辺式")):
    isAddressAssign = True
if isAddressAssign:
 left = exp.child("左辺式").node_exp_variable()
       .child()[0]
 ref_left = left.reference_node(func)
 if ref_left.ancestor("関数定義"):
  if ref_left.parent("引数リスト"):
    print"適切な記憶期間ではないコードである"
  else :
    print"適合コードである"
 else :
  print"適切な記憶期間ではないコードである"
else :
 print"適合コードである"
```

この問い合わせに記述によって，たとえば，「自動記憶期間の変数の値を静的記憶期間の変数へ代入」しているコードの有無を検査することができる．

4.3　検査時間の短縮

4.1 で述べたように，検査のためには，a) 構文要素やコードパターンを抽出するための問い合わせ記述と，b) 抽出されたコードが（検査項目によって指定された）条件を満たすかどうかを判定するための問い合わせ記述とが必要になる．これまでは，検査項目によっては，a) の問い合わせ記述の出力結果を目視し，b) の条件判定に必要とされる情報を手作業で取り出す必要があった．それが，API の改良により，a) と b) を１つの問い合わせ記述とすることが可能となり，検査時間の短縮が実現された．

5　おわりに

本研究では，CERT-C の検査項目の記述をとおして，CXmlPyQuery がもつ API の改良を試みた．その結果，これまで１つの問い合わせ記述によって検査できなかった項目が記述可能となり，検査時間の短縮がはかられた．今後の課題としては，CERT-C 以外の検査項目の記述可能性の検討などがあげられる．

参考文献

[1] 高橋耶真人, 福原和哉, 猪股俊光, 新井義和, 今井信太郎. パターン照合を用いた対話型静的検査ツールの開発. 電子情報通信学会ソサイエティ大会, A-9-1 (2012).

[2] 福原和哉, 高橋耶真人, 猪股俊光, 新井義和, 今井信太郎. 組込みソフトウェア向けコーディング規約チェッカのためのカスタマイズの一方式. FIT2012 第 11 回情報科学技術フォーラム, c-024 (2012).

[3] 岩間恵梨沙 , 福原和哉 , 猪股俊光 , 杉野栄二 , 新井義和 , 今井信太郎 , 成田匡輝 C ソースコード静的解析のための問い合わせ 言語 CXmlPyQuery とその応用 , 信学技報, Vol. 117, No. 137, pp. 125-131 2017.

[4] ”JapanComputerEmergencyResponseTeamCoordinationCenter (CERT-C コーディングスタンダード)”, https://www.jpcert.or.jp/sc-rules/, (参照 2020-5-27).

[5] ”IronPythiopn.net”, https://ironpython.net/, (参照 2020-5-27)