クライアントサイドにおける Web ページ内のプログラム記述の閲覧機能拡張に関する研究
–HTML 文書内のプログラム断片の抽出と解析–

2008MI131 松野 秀泰　　2008MI143 森 瑞歩
指導教員　吉田 敦

1　はじめに

近年，プログラミング技術の向上，既存のコードの改善や検査を目的として Web ページ内のプログラム記述を閲覧する機会が増加している．また，閲覧者が目的に合うプログラム記述を探す際, 多くのサイトを閲覧する．プログラム記述を含む Web ページは，リポジトリサイトや解説サイトなどがある．

Web ページ内のプログラム記述は，必ずしも理解しやすいものではない．具体的には，記述者のコーディングスタイルによって，閲覧者が見栄えが悪いと感じる場合がある．また，大規模なプログラム記述では，閲覧者が必要としない関数は多く記述されている場合，閲覧の妨げになる．このような問題を解決することで，可読性が上がり，学習時間や作業時間の短縮に繋がる．

理解しやすい見栄えに変更する簡単な方法は，サーバサイドで変更することである．しかし，閲覧者によって見栄えの好みが異なるので，サーバーサイドですべての閲覧者の要求を満たす見栄えに変更できない．また，閲覧者がサーバサイドの見栄えを変更できない．別の手段として，閲覧者が既存の理解支援ツールや環境 [1] を用いる方法がある．しかし，ツールや環境で理解支援するには，閲覧者の PC で既存のツールをインストールしたり，1 つ 1 つ Web ページ内からプログラム記述をダウンロードする手間がかかる．また，プログラム記述をダウンロードすることで，HTML 文書のタグなどで付加された情報を利用できなくなり，閲覧者の理解に重要な情報を失う場合がある．

より多くの閲覧者の要求を満たし，既存のツールでの手間を除くために，クライアントサイドで bookmarklet を利用してプログラムの理解支援機能を挿入することを考える [3]．そのような理解支援をするためには Web ページ内のプログラム断片を構文解析する必要がある．構文解析をするためにはプログラム断片の範囲の特定と，どのような理解支援がされるのか限定できないので，タグを用いて表現されている見栄えを維持するためにタグを含んだ構文解析木を構成する必要がある．

クライアントサイドで理解支援をするためには Web ページ内のプログラム断片を構文解析する必要があり，構文解析する上で非プログラム記述とプログラム記述の区別がついていないのでプログラム断片の範囲の特定が難しい．また，タグの扱いを考慮した構文解析器が存在しない．これらの問題点を解決するための方法を提案する必要がある．

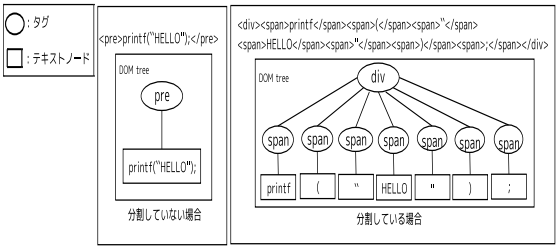

図 1　タグの入れ方の例

本研究では，クライアントサイドで bookmarklet を利用してプログラムの理解支援機能を挿入するために，HTML 文書からプログラム断片を抽出し，抽出したプログラム断片の構文解析を目的とする．Web ページ上のプログラム断片の構文解析するためにプログラム断片の範囲を特定する必要があり，非プログラム記述とプログラム記述を区別し，プログラム断片を抽出する方法とタグを含んだ構文解析木を構成するためにプログラム断片内にタグを含むプログラム記述に対して正確な構文解析をする方法を提案する．本研究で抽出また，解析するプログラミング言語は学習や開発に幅広く使用されている C 言語を対象とする．

2　抽出と解析の課題

2.1　抽出の課題

HTML 文書内のプログラム断片の抽出をする際に生じる技術的課題を以下に挙げる．

- プログラム記述と非プログラム記述を区別する基準が必要
- 抽出する境界の決定が必要

HTML 文書内のプログラム記述は DOM tree[2] の葉のテキストノードに記述されている．テキストノードにはプログラム記述だけでなく，非プログラム記述も記述されている．プログラム記述の特徴をみつけだし，これらを区別する基準を決定する必要がある．また HTML 文書内のプログラム記述は，HTML 文書の作成者によって記述の仕方が様々である．例えば，“`printf("Hello");`” というプログラム記述は HTML 文書内では，図 1 に示したようにプログラム記述内にタグを入れ分割して記述する作成者と，タグを入れずにまとめて記述する作成者がいる．プログラム記述内にタグが入っている場合，プログラムが分割して記述されている．よって，タグが入っていない場合と比べて抽出する境界の決定が難しくなる．第 3 章で抽出方法を 2 つ提案

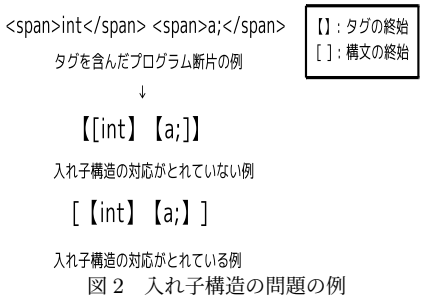

図 2　入れ子構造の問題の例

し，これらの課題を解決していく．

2.2　解析の課題

抽出したプログラム断片の書換えを支援する環境として，TEBA[4] が提案されている．タグを含んだプログラム断片の構文解析には問題点が 2 つあり，以下に示す．

- 正確な解析情報を与れられない
- 入れ子構造の対応がとれていない

最初の問題は，正確な解析情報が与えられないという問題である．TEBA を拡張しタグを字句として定義した場合，字句の並びから推測を行い情報を与えている ID_FUNC や ID_VER などの情報が HTML 文書内のタグを含んだプログラム断片では IDENT となる．

2 つ目の問題は，入れ子構造の対応がとれていないという問題である．プログラム記述に対して機能挿入する際に，DOM tree を構成する．DOM tree を構成する上でタグを含んだ構文木の構成が必要である．図 2 は入れ子構造の対応付けを示した例である．入れ子構造の対応をとるために構文の始まりと終わりの間にタグが含まれている必要がある．

3　プログラム断片の抽出

本章では抽出の課題を解決するために，プログラム記述と非プログラム記述を区別する基準を決定する．また，プログラム記述は DOM tree の葉に記述されていることから DOM tree を後順にたどり，区別する基準を用いた，プログラム記述と非プログラム記述の境界を決定した 2 つの方法を提案する．

3.1　プログラム記述の特徴

プログラム断片を抽出するためには，プログラム記述と非プログラム記述を区別する基準を決定する必要がある．プログラム記述と非プログラム記述を比較することでプログラム記述の特徴を見つけ出し，それをもとに基準を決定する．HTML 文書内のプログラム記述と非プログラム記述の頻出するトークンを調査した結果の一部が表 1 と表 2 である．次に，Web 上の C 言語が記述されたページの自然言語のテキストノードに対し頻出するトークンの調査を行った．表 2 が調査結果の一例である．これらの表から，プログラム記述には記号と予約語が頻出することがわかった．よって全体のトークン数の中で記号と予約語の占める割合が高いほどプログラム記述である可能性が高くなる．“記号と予約語のトークン数/全体のトークン数” を点数とし，これをプログラム記述と非プログラム記述を区別する基準とする．この点数を用い，プログラム断片の境界を決定する 2 つの方法を提案する．

表 1　C 言語の調査結果

トークン数	1	回数	2	回数	3	回数
206	;	31	=	16	/*	11
340	;	33	=	14	int	8
5581	;	1072	=	614	int	261
6809	;	462	)	297	=	230
129	/*	4	*/	4	;	2

表 2　非プログラム記述の調査結果

トークン数	1	回数	2	回数	3	回数
4060	the	285	of	131	to	122
9575	the	489	of	263	to	253
6200	the	243	is	170	a	165
1222	the	54	you	42	to	34
4569	the	254	to	154	a	110

3.2　抽出方法 1

HTML 文書内のプログラム記述に多く使用されるタグを境界にした抽出方法を検討した．

3.2.1　HTML 文書の分析

HTML 文書内のプログラム記述は DOM tree の葉のテキストノードに記述されている．Web 上のページからプログラム記述を含むページを，ランダムに 40 件選びプログラム記述のテキストノードがどのような位置に記述されているのかを調査した．その結果プログラム記述のテキストノードは特定のタグの子や孫に位置する場合が非常に多かった．調査の結果から，プログラム記述が記述されているテキストノードの位置について分類した．

分類 A　PRE タグの子のテキストノード
分類 B　BODY タグの孫のテキストノード

ここで BODY タグの子が PRE タグの場合は分類 A に含むとする．調査した Web ページ 40 件のうち 28 件が分類 A，9 件が分類 B であった．残りの 3 件はプログラム記述のテキストノードの位置がばらばらであり件数が少ないので分類をせずその他とする．これらの分類の位置にはプログラム記述だけでなく非プログラム記述が記述されている場合がある．よって，分類だけではプログラム断片の抽出はできない．テキストノードに着目し，プログラム記述なのか非プログラム記述なのかを機械的に区別する必要がある．そこで 3.1 節の点数が高ければプログラム記述であると判別する．

3.2.2　抽出方法の説明

HTML 文書内のプログラム記述の位置に着目した抽出方法を提案する．分類 A，分類 B の位置にテキスト

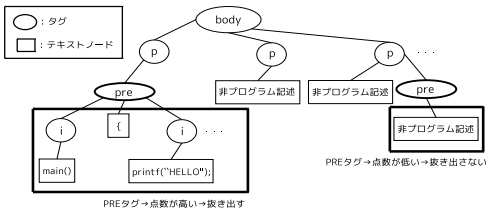

図 3　抽出方法 1

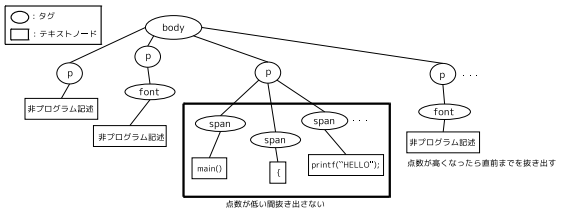

図 4　抽出方法 2

ノードがあった場合，その子や孫のテキストノード全部をまとめて計算した 3.1 節の点数が高ければ，図 3 のように子と孫のテキストノードのすべてを抽出することでプログラム断片を抽出する．

3.3　抽出方法 2

抽出方法 1 で着目したテキストノードの位置は HTML 文書の作成者に大きく依存する．よって，テキストノードの位置に依存せず，3.1 節の点数の変化を境界とした方法を提案する．

DOM tree を後順にたどり点数が低い間 DOM tree をたどり，点数が高くなった場合そこを境界として，図 4 のようにたどってきたノードをプログラム断片として抽出する．これにより分割して記述されたプログラム記述をまとめて抽出できる．

4　抽出したプログラム断片の解析

4.1　属性付き字句系列の利用

HTML 文書に存在するプログラム断片は関数のみのプログラム断片など不完全なプログラム記述が存在する．これらの不完全なプログラム記述に対して有効な解析器が TEBA である．TEBA の解析結果は “種別, (識別番号), <テキスト>” の順に出力される．種別は予約語や演算子など字句の種類を示す．また，識別番号とは括弧の対応関係を表現するために示されており，種別の直後に識別番号がつけられる．テキストとはプログラム記述のその字句に対して対応しているテキストを示す．

4.2　解決策の提案

タグを含んだプログラム断片に対して正しい解析結果を得るための方法を提案する．正しい解析結果の “正しさ” には 2 つの観点があり，以下に示す．

- タグを含むプログラム断片を TEBA で解析する際にタグの前後に存在する字句に対して与えられる種別が正しいこと
- 入れ子構造の対応がとられていること

タグを含むプログラム断片を TEBA で解析する際に予約語や変数などの種別について正しい解析結果を得るために，タグの種別を定めることでタグの前後に存在する字句に対して正しい種別を与える．また，入れ子構造の対応をとるために，プログラム断片内に含まれる開始タグと構文の始まりを示す仮想字句，構文の終わりを示す仮想字句と終了タグを入れ替える．

4.2.1　正しい解析結果を得るためのタグの種別

構文解析においてタグは不要であるので，タグを無視するためにスペース字句として扱う方法を提案する．タグをスペース字句として扱うと，プログラム断片内のタグを無視することができ，タグの前後にある字句に対して正しい種別を与えられる．タグをスペース字句として扱う際の種別において，開始タグを SPACE_TB，終了タグを SPACE_TE と定義する．

4.2.2　入れ子構造の問題について

入れ子構造の対応をとる方法として，開始タグと構文の始まりを示す仮想字句と終了タグと構文の終わりを示す仮想字句の入れ替えを提案する．構文の始まりと終わりを示す仮想字句はスペース字句を無視し、字句情報が与えられる．本研究では，正しい解析結果を得るためにタグをスペース字句として扱っているので，構文の始まりを示す仮想字句の後にタグ，構文の終わりを示す仮想字句の前にタグは存在しない．よって，開始タグの後に構文の始まりを示す仮想字句，終了タグの前に構文の終わりを示す仮想字句が存在する場合，開始タグと構文の始まりを示す仮想字句，構文の終わりを示す仮想字句と終了タグを入れ替える．

TEBA で構造体を構文解析すると 2 重の入れ子構造が生じる．2 重の入れ子構造とは，TEBA で構造体のプログラムを解析すると構文の始まりを示す仮想字句が 2 つ並んで解析結果として得られることを意味する．2 重の入れ子構造が起きるのは共用体，構造体，列挙型の 3 つの場合である．構文の始まりを示す仮想字句が 2 つ並んで出力された場合，開始タグと構文の始まりを示す仮想字句を 2 回入れ替える．また，構文の終わりを示す仮想字句と終了タグも入れ替える．

実装として，TEBA の syntax.rules を拡張した．Perl で記述し，正規表現を用いて，開始タグと構文の終わりを示す仮想字句の入れ替え，また構文の終わりを示す仮想字句と終了タグを入れ替える．

5　検証・考察

5.1　抽出の検証

本研究で提案した抽出方法について検証する．

5.1.1　評価方法

抽出方法 1 と抽出方法 2 を検証するために，3.1.1 節の分類 (分類 A，分類 B，その他) とプログラム記述内のタグの有無を考慮し検証する．分類 A，分類 B，その他のそれぞれに対し，タグがある場合とない場合の HTML 文書を見つけた．しかし，60 件の中から見つけたところ，分類がその他でタグがないページが 1 件も存在しな

表 3　評価方法

“抽出できた” が一番多い場合	○
“余分があるが抽出できた” が一番多い場合	△
“抽出できない” が一番多い場合	×

表 4　抽出方法 1 の評価結果

分類	タグ	評価
分類 A	有	○
分類 A	無	○
分類 B	有	△
分類 B	無	○
その他	有	×

表 5　抽出方法 2 の評価結果

分類	タグ	評価
分類 A	有	△
分類 A	無	○
分類 B	有	△
分類 B	無	△
その他	有	○

かった．よってこの条件は検証対象としない．分類 A，分類 B のプログラム記述内にタグがある場合とない場合の HTML 文書をそれぞれ 3 件，その他のプログラム記述内にタグがある HTML 文書を 3 件，計 15 件を検証対象とする．評価方法は表 3 に示す通りである．

5.1.2　検証結果と考察

検証の結果を表 4 と 5 に示す．評価結果から分類 A，2 の場合方法 1 の方が適しているが，タグの位置による抽出方法では限界がある．よってより多くの Web ページ内のプログラム断片を抽出するには方法 2 の抽出基準を増やし，精度の向上が必要であることがわかった．

5.2　解析の検証

HTML 文書のタグをスペース字句として扱うように TEBA を拡張することで，タグを含むプログラム断片の構文解析が可能であることを確認する．また，入れ子構造の対応がとれているのかを確認する方法は目視で行う．

5.2.1　HTML 文書の調査・分類

プログラム断片内にタグが存在する場合，タグの位置は HTML 文書によって異なるので，タグを含むプログラム断片 40 件に対して調査し，分類した．

分類 1　タグの位置が行末毎
分類 2　タグの位置が 1 行毎
分類 3　タグの位置がトークン毎

分類結果として，断片数が 14 でタグの位置が行末毎の場合を分類 1，断片数が 1 でタグの位置が構文を無視している場合が分類 2，断片数が 25 でタグの位置が構文を考慮している場合を分類 3 とする．

プログラム断片の解析の評価方法の手順を以下に示す．

手順 1　ブラウザに表示されているプログラム記述と HTML 文書内のプログラム記述を構文解析
手順 2　構文解析した 2 つのプログラム記述の解析結果から diff コマンドを用いて差分を抽出

HTML 文書にはタグと特殊文字が存在する．特殊文字とはブラウザ上で “<” を示す “`<`” や “>” を示す “`>`” などを意味する．タグと特殊文字を含んだ字句が差分として抽出されることでタグを含むプログラム断片に対して構文解析が可能であることがいえる．

5.2.2　検証結果と考察

分類 1，分類 2，分類 3 ではタグと特殊文字を含む字句が差分として抽出されたので正確な解析情報が付与された．また，分類 2 を除いて構文木を構成するための入れ子構造の対応がとれた．

考察として，タグをスペース字句として扱うように TEBA を拡張することで本研究で調査した分類においては，タグを含むプログラム断片に対して構文解析が可能であることがわかった．また，構文木を構成するための入れ子構造の対応については開始タグと構文の始まりを示す仮想字句，構文の終わりを示す仮想字句と終了タグを入れ替えることで入れ子構造の対応がとれた．分類 2 の入れ子構造の対応がとれなかった理由として，分類 2 ではタグの位置が構文を無視しているからである．タグの位置が構文を無視しているので，タグを取り除く方法以外では入れ子構造の対応をとることが難しい．

6　おわりに

本研究では HTML 文書の構成の特徴と C 言語のプログラム記述の特徴を利用して，HTML 文書内のプログラム断片を機械的に抽出した．機械的に抽出する上での問題点を C 言語のプログラム記述の特徴とテキストノードの位置を考慮する方法でプログラム断片の抽出が可能になった．また，抽出した中でタグを含んだプログラム断片の構文解析は解析器である TEBA の拡張によって可能になった．

今後の課題として，余分に抽出してしまった記述を少なくするために C 言語のプログラム記述に頻出するトークンを調査し，抽出基準を設け，検討する必要がある．

参考文献

[1] Eclipse, “*Eclipse - The Eclipse Foundation open source community website.*,” http://eclipse.org/, 2011.

[2] W3C, “*World Wide Web Consortium (W3C)*,” http://www.w3.org/, 2011.

[3] 森島敦子，椎名優貴，土屋陽平，“クライアントサイドにおける Web ページ内のプログラム記述の閲覧機能拡張に関する研究 - プログラム記述への理解支援機能の挿入方法 - , ” 南山大学 数理情報学部 情報通信学科 2011 年度卒業論文要旨集.

[4] 吉田敦，蜂巣吉成，沢田篤史，張漢明，野呂昌満，“属性付き字句系列に基づくプログラム書換え支援環境の試作，” ソフトウェアエンジニアリング最前線 (ソフトウェア・エンジニアリング・シンポジウム 2010 予稿集)，pp.119-126，Aug. 2010.