プログラムのパターン変換系における空白字句保存手法に関する研究

2007MI192 大島誠也
指導教員：吉田敦

1 はじめに

ソフトウェアの開発等で，プログラムを記述するときに，一度ですべて書ききることは難しく，仕様を満たすように小規模な変更を積み重ねていく必要がある．また，一度完成したプログラムでも，運用，保守の段階でプログラムに仕様変更等が生じるので，プログラムの書き換え作業を何度も繰り返す必要がある．このような書き換え作業の中では，コードクローンに対する修正や関数呼出しなどの識別子の参照に対する変更など，分散する複数の箇所に同じ編集作業をする場合がある．このような編集作業は，プログラムの書き換えツールを用いることで，手作業で単純作業を繰り返すことで起こる人為的ミス混入を回避でき，作業効率の向上に貢献する．既存の書き換えツールとしてプログラムのパターン変換ツール [1][2] がある．

プログラムの書き換え支援では，書き換え後も継続的にプログラマがそのプログラムを読み，編集もしていくことから，処理の把握に役立つコメントやプログラムの可読性を向上させる空白・改行を保存することが不可欠である [3]．しかし，既に提案されている書き換え支援環境である TEBA[1] や TXL[2] ではコメントや空白を保存する仕組がなく，書き換え対象箇所のコメントや空白の欠落が生じる．

パターン変換でコメント・空白・改行を表す空白字句を維持する方法として，空白字句を一つの構文要素として扱い，書き換え前の空白字句を書き換え後のどこの箇所に維持させるかを明示的に書くことが考えられる．しかし，この方法をプログラマが手作業で行うには，パターンが複雑化し，記述の労力も大きい．この処理は書き換えの本質的な作業ではないので，この作業を自動化することが望ましい．

関連研究としてプログラム書き換えツールから独立したスタイル維持方法に関する研究 [4] があるが，この研究では，書き換えるプログラム全体の空白字句をまとめて維持するので，条件によっては書き換えられていない箇所の空白字句が維持されない．本研究は，プログラムのパターン変換を行うときに，空白字句の書き換え後の位置を決定し，自動的に挿入する仕組を提案する．また，本研究では実装が公開されていて，かつ変更しやすいことから TEBA を対象に拡張を行う．ただし，基本的な処理は他の変換系にも適用可能である．

2 背景技術

2.1 属性付き字句系列

プログラムの書き換え支援環境 TEBA は，ソースプログラムを字句解析して得た字句系列に対して構文上の種別などの情報を与えることで，抽象構文木と同等の情報を持つ属性付き字句系列を生成する．この属性付き字句はテキストを結合することで元のソースプログラムを再現できる．TEBA はソースプログラムを解析して得た属性付き字句系列を書き換えることによってソースプログラムの書き換えを実現している．

2.2 プログラムのパターン変換

TEBA が提供するプログラムのパターン変換は，書き換え前と後の 2 つのソースコード断片を変換前パターンと変換後断片として記述したパターン変換記述に従い行う．パターン変換記述とは，変換前パターンに記述したプログラム断片が対象のソースプログラムに含まれると，変換後断片に記述したプログラム断片に書き換えるという情報を記述したものである．パターン変換記述には，パターン変数という差異のある要素を可変部分として抽象化する仕組の字句があり，変換前の指定された構文要素に適合して，変換後の参照箇所に適合した構文要素をあてはめ維持する．パターン変数は，変換前パターンでは`${名前:型}`と記述し，変換後断片では`${名前}`と記述する．本研究は，空白字句の構文要素に適合するパターン変数 (以下，空白パターン変数) を用意して，字句間の空白字句を維持する．

3 空白字句保存手法

パターン変換に使用するパターン変換記述に空白字句を保存する記述を加える補正を行うことで，書き換え後のプログラムに空白字句を維持する方法を提案する．また，書き換え後のプログラムに対しても，余分な空白や改行によりインデントが乱れないように処理を行う．

3.1 パターン変換記述の空白字句保存補正

本研究では，パターン変換記述を字句系列の変換ルールに生成するときに空白字句保存補正を行う．補正内容は，パターン変換記述の変換前パターンと変換後断片の対応する字句同士に空白パターン変数を付随させることで，書き換え後プログラムに空白字句を維持する処理である．対応する字句はパターン変換記述に明記されていないので，LCS(Longest Common Subsequence) のアルゴリズムで共通字句系列を求めたときに対応したと判定した 2 つの字句の関係とする．この補正処理の全体像を図 1 に示し，各処理について説明していく．

3.1.1 パターン変換記述の解析と補正処理

パターン変換記述は TEBA で解析し，変換前パターンと変換後断片の字句列に分ける．LCS で精度の高い対応関係を得るために，変換前パターンと変換後断片の字句

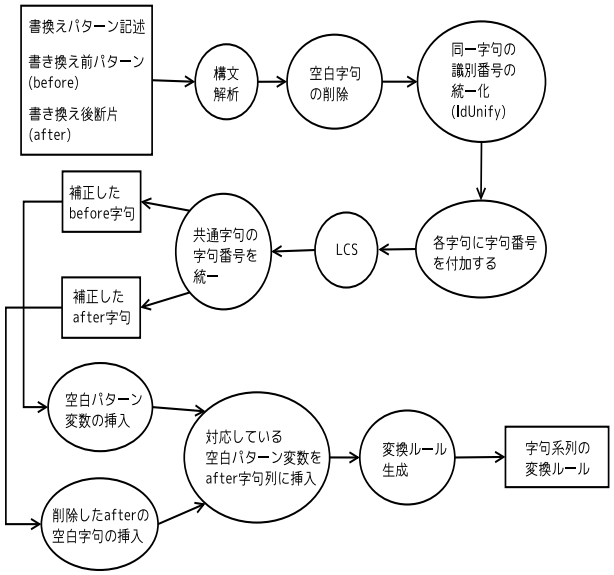

図 1 空白字句保存補正を行うツールの全体像

列に含まれる空白字句の削除と IdUnify ツールで対応する各字句列の識別番号を統一する．

3.1.2 字句番号の付加と LCS

各字句列に相違する字句番号を付加させ，付加した字句番号を無視した LCS を求める．LCS で得た対応字句の字句番号を統一し，字句の対応関係を表現する．

3.1.3 空白パターン変数と空白字句の挿入

変換前パターンの字句間には名前が相違する空白パターン変数を挿入し，変換後断片の字句列には 3.1.1 節で削除した空白字句を復元する．空白字句を復元する理由は，削除した状態のままでは，識別子と識別子の間など，本来，1 つ以上存在すべき箇所の空白が欠落し，書き換え後プログラムにエラーが生じる可能性があることである。

3.1.4 変換後断片への空白パターン変数の挿入

変換後断片への空白パターン変数は，3.1.2 節で得た変換前パターンの対応字句に付随させる方法で，変換前パターンの空白パターン変数をすべて挿入する．空白パターン変数が付随する字句は，前後に存在する対応関係がある字句の中で最も近くにある字句である．前後にある対応する字句の距離が同じ場合は，前にある字句を優先する．

3.2 空白字句の補正

3.1.3 節で復元した空白字句と変換前パターンから保存した空白字句を両方維持することで書き換え後プログラムのインデントが乱れる可能性がある．この問題を軽減させるために，書き換え後プログラムの同一字句間に変換前プログラムと変換後断片の空白と改行が存在したら，変換後断片の重複した空白と改行を削除する補正を加える．

4 評価・考察

3.1 節で提案した空白字句保存補正の妥当性を確認するために実装を行い，複数の事例に適用した．適用した例として，パターン変換の記述例を図 2 に，入力したプログラムと変換後の出力を図 3 に示す．図 2 は，for 文を while 文に書き換える．図 3 を見ると，各字句間にコメントを挿入して変換を行ったが，すべてのコメントが維持できている．また，字句の移動も含む書き換えだが，移動した字句にも前後のコメントが付随していることも分かる．

```
% before
for ( ${init:EXPR} ; ${cond:EXPR} ; ${succ:EXPR} )
  ${stmt:STMT}$;
% after
${init};
while (${cond}) {
  ${stmt}$;
  ${succ};
}
% end
```

図 2 パターン変換記述例

```
/*書き換え前*/
  for /*cmt1*/ ( /*cmt2*/ i = 1 /*cmt3*/ ;
/*cmt4*/ i <= 10 /*cmt5*/ ; /*cmt6*/ i++ /*cmt7*/ )
{ /*cmt8*/
    sum = sum * i; /*cmt9*/
    printf("%2d =  %7d\n", i, sum); /*cmt10*/
  }
```

```
/*書き換え後*/
   /*cmt1*/  /*cmt2*/ i = 1 /*cmt3*/ ; /*cmt4*/
while (i <= 10 /*cmt5*/ ) {
  { /*cmt8*/
    sum = sum * i; /*cmt9*/
    printf("%2d =  %7d\n", i, sum); /*cmt10*/
  }
   /*cmt6*/ i++ /*cmt7*/ ;
}
```

図 3 ソースプログラムの書き換え例

5 おわりに

本研究では，空白字句を保存する手法を提案・実装し，実現できることを示した．今後の課題は，複数のパターン変換例の検証を通して空白字句の付随方法を洗練することである．

参考文献

[1] 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ソースコード書き換え支援環境，” 情報処理学会論文誌，vol.53，no.7，pp.1832-1849，July 2012.

[2] “*The TXL Programming Language*,” http://www.txl.ca/, 2012.

[3] Waddington, Daniel G. and Yao, Bin, “High-Fidelity C/C++ Code Transformation,” Electron. Notes Theor. Comput. Sci., vol.141, no.4, pp.35-56, 2005.

[4] 本田竜二，石原浩佑，立道昂太，“プログラム書換えツールから独立したスタイル維持方法に関する研究，” 南山大学 数理情報学部 情報通信学科，2010 年度卒業論文要旨集.