修士論文進捗
(2007/1/9)

M2 佐藤秀明

実現したいこと

- コードクローンを解消したい
  - ソースコード中の似ている部分を一つにまとめる
  - 大規模システム開発のメンテナンス性向上が目的

コードクローンの定義

- 「互いに等しいトークン列の組」と定義
  - ただし変数のリネームについては等しいとみなす

```
int f(int x){
    int a = 3;
    return a * x;
}
```

クローン

```
int g(int y){
    int b = 3;
    return b * y;
}
```

コードクローン発見の既存手法

- 接尾辞配列を利用 [Basit et al. 2005]
  - テキスト検索の手法を応用してクローンを発見
- 接尾辞配列：文字列の全接尾辞を辞書順にソート
  - 省スペース
  - 単純な構造

actaat$

index	pos	実体
0	3	aat$
1	0	actaat$
2	4	at$
3	1	ctaat$
4	2	taat$
5	5	t$
6	6	$

既存手法の問題

- 見つけたクローンを報告するだけ
  - 書き換えはプログラマが自分でやるしかない
- 書き換えまでサポートすると解析時間が膨大に

より高速な解析の必要性

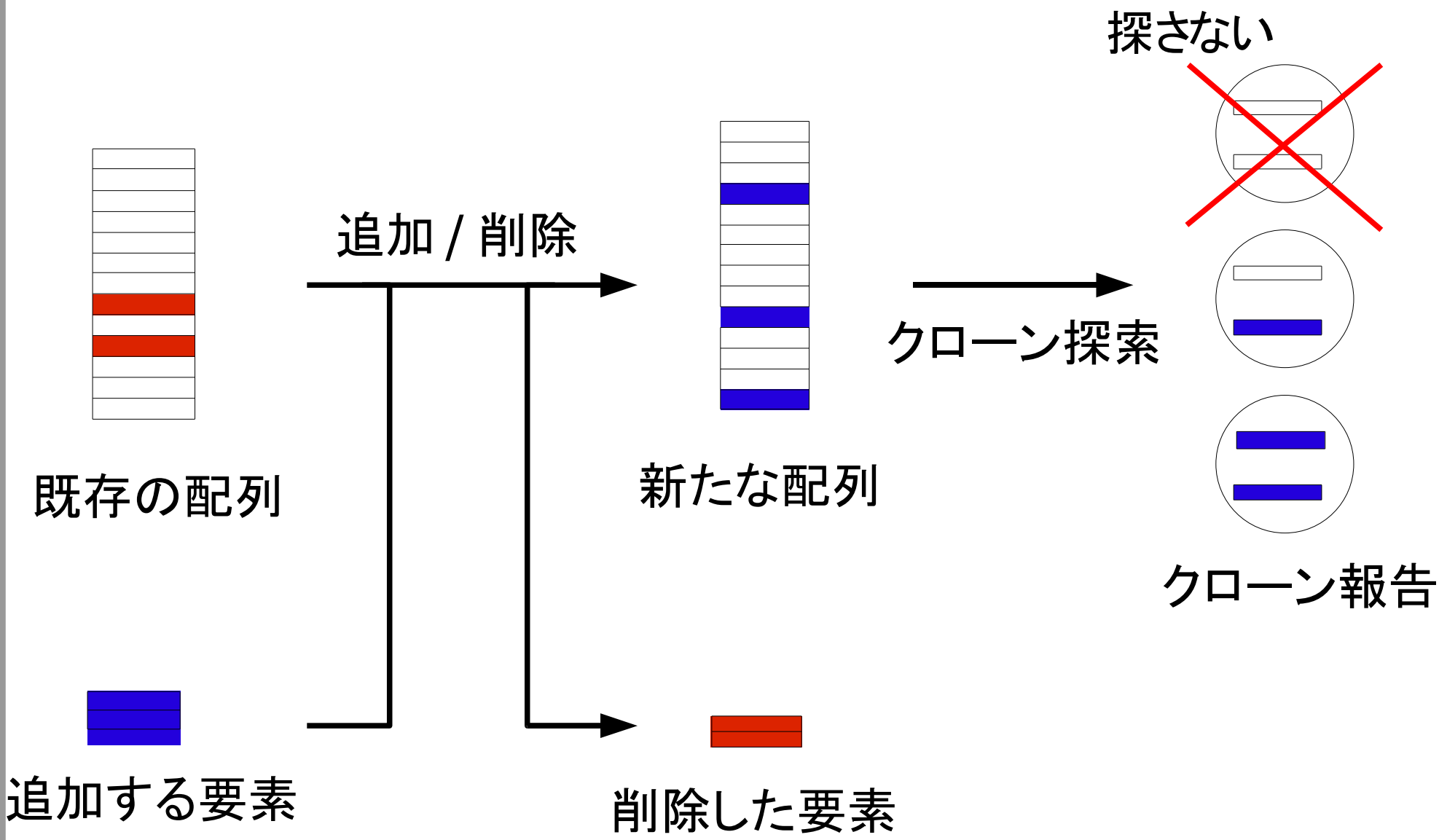
我々のアプローチ

- ソースコードの変化に沿って解析
  - 差分のみ解析することで毎回の計算量を削減
  - コードを常時きれいに保てる

解析の方針

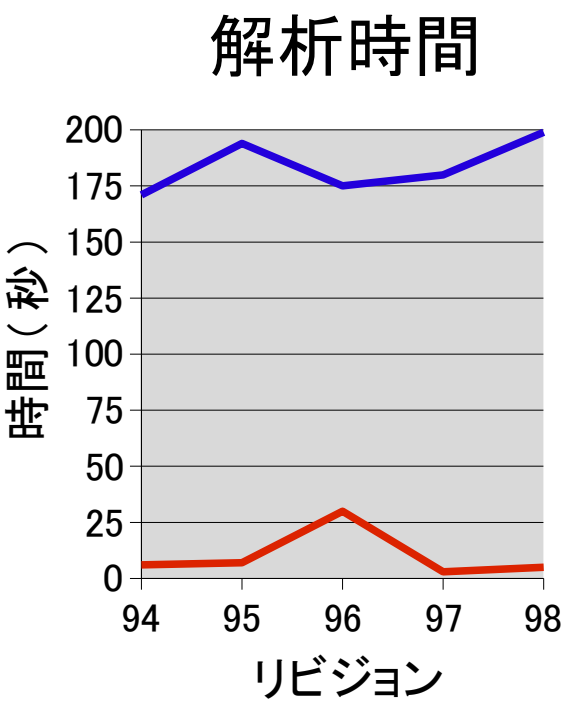
- 前回作った接尾辞配列に今回生まれた差分を反映
  - 毎回全てを作り直すより速い
- 今回新たに生まれたクローンのみ発見
  - 調べるべきクローンの組み合わせを減らす

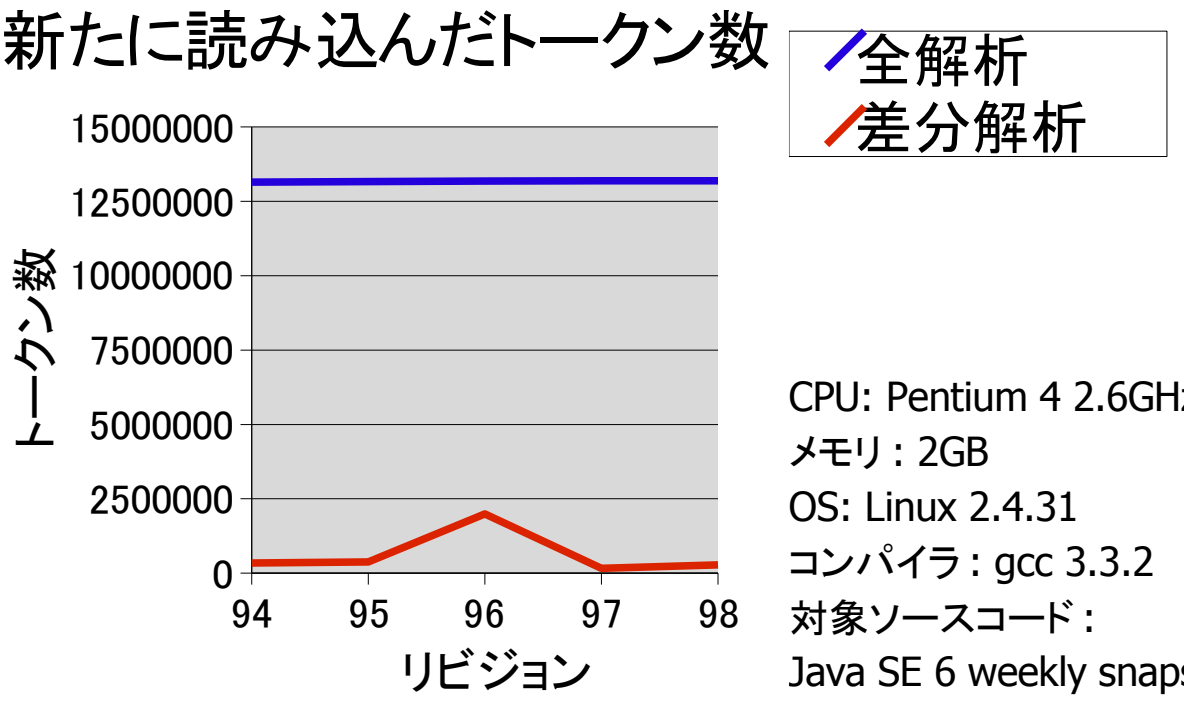
解析手順



中間成果

- クローン発見のための解析時間を削減
  - おおむね 95% 程度の時間短縮





CPU: Pentium 4 2.6GHz
メモリ：2GB
OS: Linux 2.4.31
コンパイラ：gcc 3.3.2
対象ソースコード：
Java SE 6 weekly snapshot

進捗

- クローン発見アルゴリズムの改良
  - 計算量の改善
- 3 要素以上からなるクローンを報告可能に
  - 今までは 2 要素からなるクローンしか報告できず
- C から OCaml へのインターフェイス構築
  - クローン発見は C(速度重視)
  - ソースコード変換は OCaml(変換の正しさ重視)

いまやっていること

- 見つけたクローンを関数として抽出
  - ソースコード変換の候補としてユーザへ提示
- 今週中にはできそう

予定

- 今週：ソースコード変換
- 来週以降：評価
  - 解析時間の測定
  - 変換例の選定
- それらと並行して：論文執筆

まとめ

- 逐次的手法によるコードクローンの高速な解析
  - 接尾辞配列の差分構築
  - 解析するクローン候補数を絞る