オープンソースと関数型言語による
プログラム解析と
カバレッジテスト自動実行

有限会社ITプランニング
今井 敬吾

第12回組み込みシステム技術に関するサマーワークショップ
@ ホテル日航豊橋

テスト手法 2つのアプローチ

* **ブラックボックステスト**
  * プログラムの入出力に着目
  * 尺度：「外部仕様通りに入出力される」こと
* **ホワイトボックステスト**
  * プログラムの内部構造に着目
  * 尺度：「プログラムの全部を通る」こと
    e.g. 命令カバレッジ、分岐カバレッジ、パスカバレッジ

課題：
自動化によるコスト低減
網羅性の向上

テストケースの自動生成問題

* ブラックボックステスト：

  ○比較的平易に実装可能

  ×外部仕様をツール入力可能な形で準備する必要あり

  (自動化の限界)

* ホワイトボックステスト：

  ○ソースコードから生成可能

  ×高度なプログラム解析技術が必要

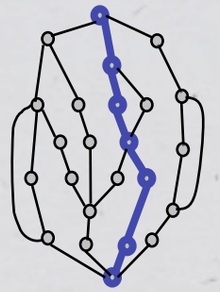
  ×数学的に自動生成困難な場合も (eg. ハッシュ関数)

最近のツール・研究で解決されてきている

(オープンソースで公開されている場合も)

プログラム解析・制約解消による テスト自動生成

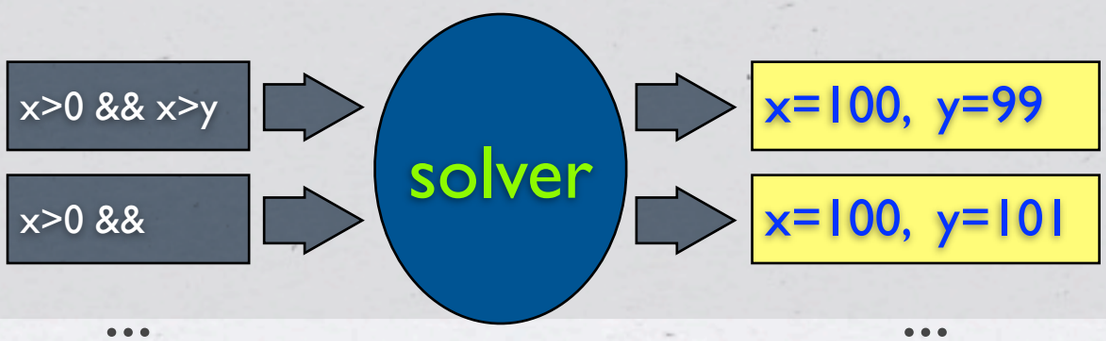
1. プログラムの分岐構造を抽出



2. 分岐条件を用い、ソースコードの各点に到達するための条件を抽出

3. 2.で求めた条件を制約ソルバーにかけて、入力値を得る

```
if(x>0) {
   if(x-y>0) {
      // x>0, x>y
   } else {
      // x>0, x<=y
   }
} else { /* x<0 */ }
```

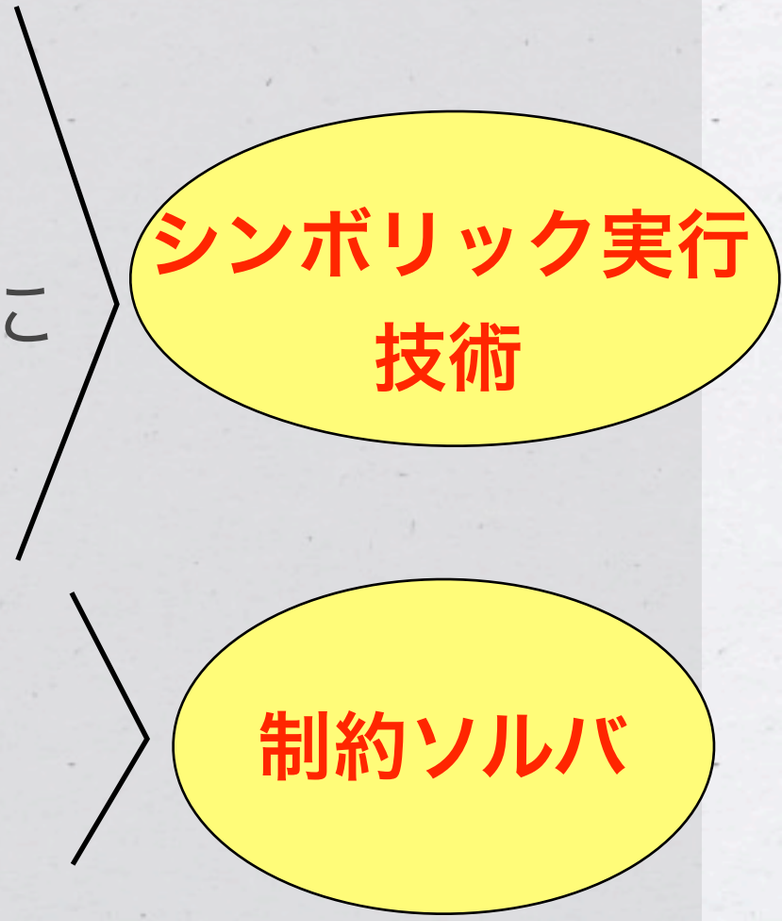


プログラム解析・制約解消による テスト自動生成

1. プログラムの分岐構造を抽出

2. 分岐条件を用い、ソースコードの各点に
   到達するための条件を抽出

3. 2.で求めた条件を制約ソルバーで解き、
   入力値を得る



2つの技術を基にしている

近年なぜ盛んなのか

(基礎的な概念は1970年代に登場)

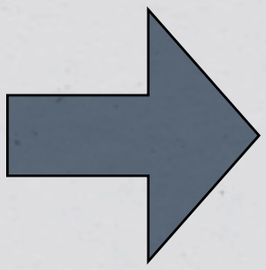
* ハードウェア性能の向上、単価の低下
* 各種プログラム解析技術の発展 **(形式手法)**
  * ソフトウェア(ソースコード)モデル検査
    : Java PathFinder, CBMC, ...
    → シンボリック実行技術に応用されている
  * 制約ソルバー
    : SATソルバー、SMTソルバー (実行効率を競うコンペ有り)

インターネットで無償入手可能なツール、文献など

- C
  - CUTE (クローズドソース)　http://osl.cs.uiuc.edu/~ksen/cute/
  - CREST (オープンソース, 未成熟)　http://crest.googlecode.com/
  - CBMC + Symbolic Execution (論文のみ)　http://www.cprover.org/cbmc/
- Java
  - Java PathFinder + Symbolic Execution (Java PathFinderに付属？)
    http://javapathfinder.sourceforge.net/
- LLVM
  - KLEE (オープンソース)　http://klee.llvm.org/

テストケースの自動生成問題: 期待できること

* ホワイトボックステスト：
  ○ソースコードから生成可能
  ○高度なプログラム解析技術 → シンボリック実行
  ○数学的に自動生成困難な場合 → コンコリック実行



より100%に近いカバレッジテストの自動化

最近の話題

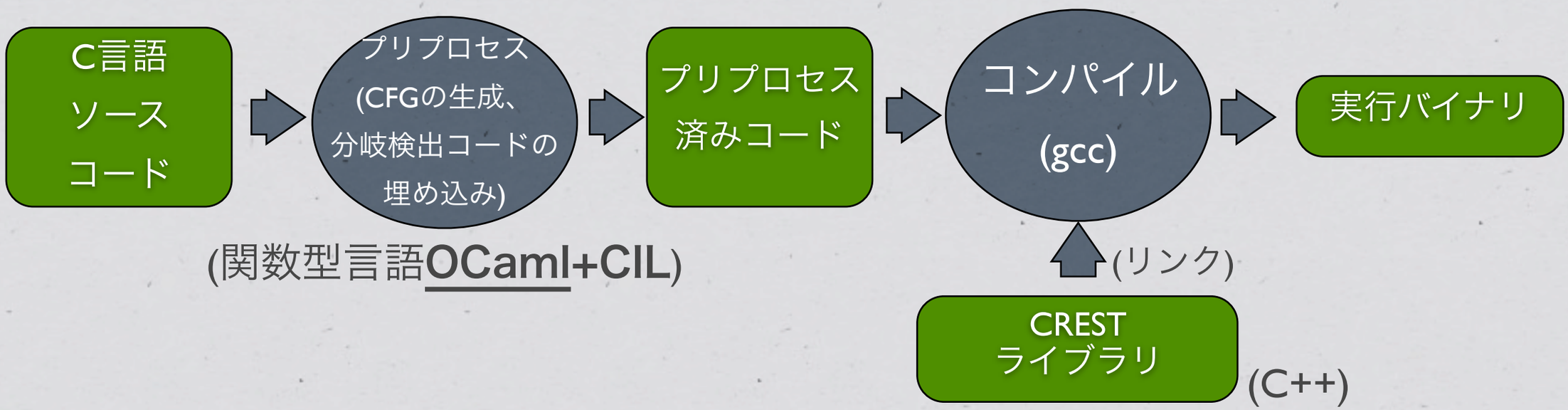
* 制約解消困難な場合における、
  具体的（非シンボリック）な実行の併用
  （ハッシュ関数など、一方向性関数は制約解消が計算量的に困難）
  * コンコリック実行(symbolic+concrete=conclic)
  * CREST, CUTEなど
* 環境依存部分の解消
  * KLEEなど

デモ：CREST

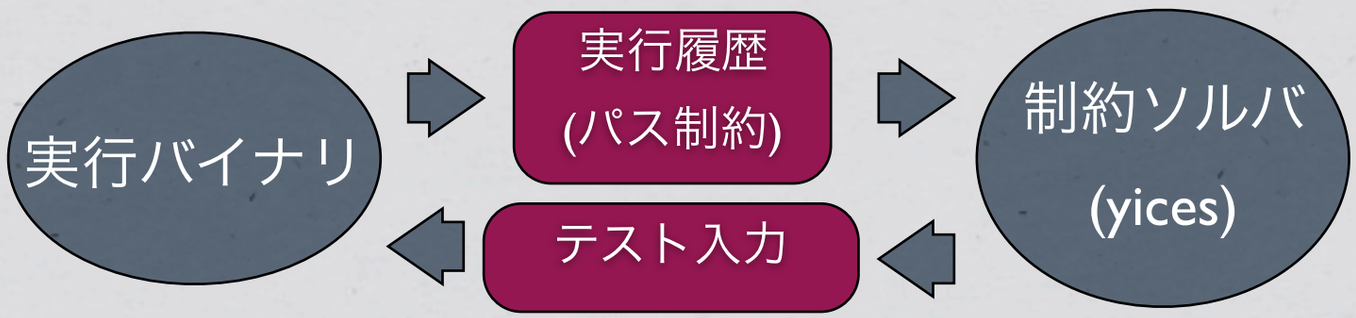
* C言語のためのカバレッジテスト自動実行ツール
  * コンコリック実行 (シンボリック実行＋通常の実行)
    * 制約ソルバ yices
* パス選択アルゴリズムを選択可能
  (深さ優先探索、CFG探索、ランダムパス選択、ランダム入力、ヒューリスティック等)
* 例1. 二次方程式の解 (CRESTのパス網羅テストの簡単な解説)
* 例2. grep (CREST付属のサンプル, 正規表現のテスト)

(参考)CRESTの内部動作

✳自動テスト用バイナリの生成 (crestcコマンド)



✳テスト自動実行 (run_crestコマンド)



繰り返し実行

分岐命令毎に生成した分岐ログを読み込み、必ず直前とは違う分岐を行うテスト入力を生成

メッセージ

* ソースコードの意味解析によるテスト自動化は有望
  * 形式手法(SATソルバ)の普及によるところが大きい
* 関数型言語はソースコード解析に向いている
  * CIL (OCaml) http://cil.sourceforge.net/
  * Language.C (Haskell) http://www.sivity.net/projects/language.c/
* 関数型言語の習得により、ソースコード解析がより身近に！
  * 筆者は実際にLanguage.C を研究で使用
    例.C89に適合するコードへの変換(一部)：http://d.hatena.ne.jp/keigoi/20080813