制約の局所性に基づく並行論理プログラム自動修正系 Kima の最適化

網代 育大　上田 和紀

早稲田大学大学院 理工学研究科
早稲田大学理工学部 情報学科

{ajiro,ueda}@ueda.info.waseda.ac.jp

概要

1. 並行論理プログラミングの簡単な紹介
2. Kima のデモ
3. Kima の動作原理
4. 変数の出現の仕方に基づく最適化
5. 制約の局所性に基づく最適化
6. 最適化による探索空間の縮小効果
7. 自動プログラミングへの応用

並行論理プログラミング

並行論理型言語の特徴

- 並行・並列・分散計算に適している。
- 論理型言語 (Prolog) からバックトラックを除いた言語。
- 計算は (並列計算/論理プログラミングにおける) リダクションによって進む。
- プロセス間通信は、論理変数を通じて暗黙のうちに行なわれる。

KL1 プログラムは以下のような節の集合：

```
p(X,Z) :- true | q(X,Y)@node(0), r(Y,Z)@node(1).
```

はじめに

Kima は強型(モード)体系を用いることによって、プログラムに関する仕様や宣言を与えることなく KL1 プログラムの軽微な誤りを自動的に修正するシステムである。

- 強型体系 (strong typing) はデータ型の一貫性を保証し、多くの簡単な誤りを静的に検出する。
- 並行論理プログラミングにおける強モード体系 (strong moding) はデータフローの一貫性を保証し、多くの簡単な誤りを静的に検出する。
- このときのモード/型解析は多数の簡単なモード/型制約の制約充足問題であり、誤りを含むプログラムでは制約集合が充足不可能 (non-well-moded) となる。

ここで デモ

1. Append プログラムの1箇所(深さ1)の誤り
2. Quicksort プログラムの2箇所(深さ2)の誤り

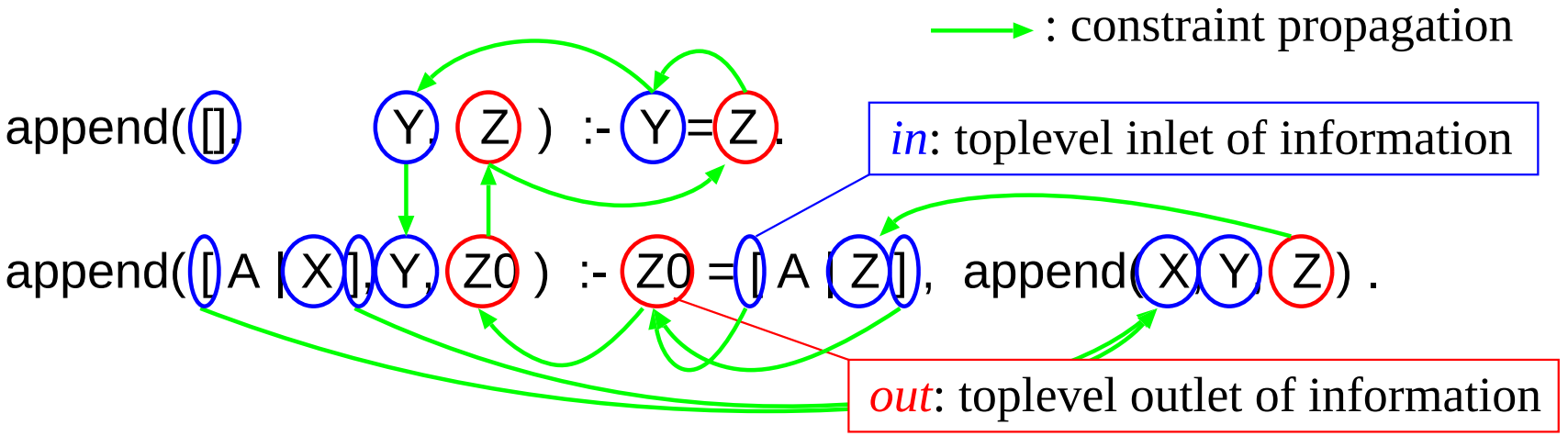
Kima の動作原理

基本アルゴリズム

1. モード/型制約の矛盾する極小部分集合の計算による、誤り箇所候補の特定
2. 以下の手続きの繰返しによる generate-and-test
   (a) 特定した場所の周辺の機械的な書換え
   (b) 書換えたプログラムのモード/型の再計算

- 計算量 = 可能な書換え × モード(型)解析の手間
- モード(型)解析の手間はプログラムサイズに比例

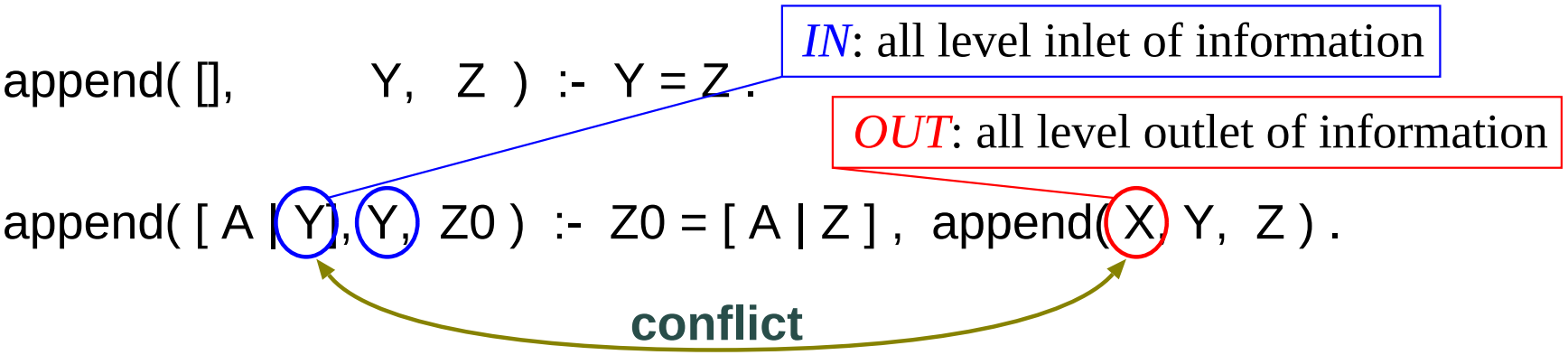
モード解析による情報の流れの解析



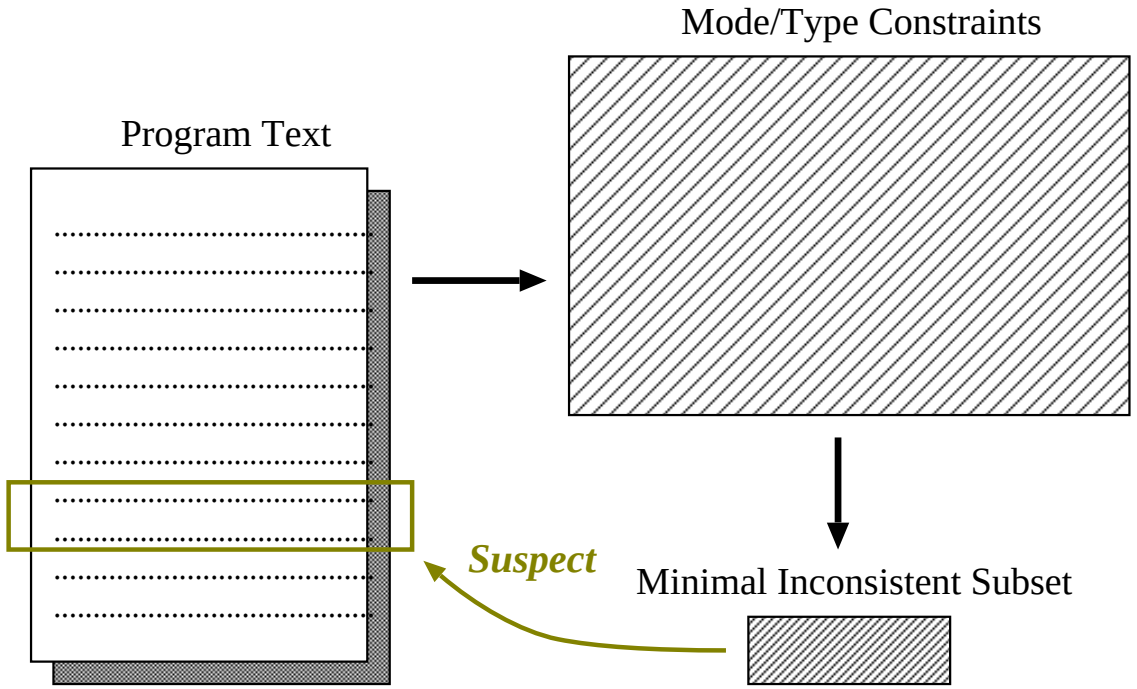
- モード解析は多数の簡単なモード制約の制約充足問題である。
- モード制約は
  - プログラムテキスト中の記号出現と
  - モードづけ規則によって課される。
- データ型に関しても同様の方法で解析可能

モード解析による誤りの検出

- 正しいプログラムはモード/型制約の集合が充足可能 (well-moded) である。
- 誤りを含んだプログラムは、制約集合が充足不可能 (non-well-moded) となることが多い。



矛盾する極小部分集合による誤り箇所の特定



- 矛盾する極小部分集合は常に間違った制約を含む。
- 極小部分集合のサイズは、プログラムサイズと関係なく小さい (およそ10 以下)。

矛盾する極小部分集合を求めるアルゴリズム

$\{c_1, \ldots, c_n\}$: a multiset of constraints (input),
S: a minimal inconsistent subset (output).

$c_{n+1} \leftarrow \mathit{false}$;
$S \leftarrow \{\}$;
while S is consistent **do**
  $D \leftarrow S$; $i \leftarrow 0$;
  while D is consistent **do**
    $i \leftarrow i + 1$; $D \leftarrow D \cup \{c_i\}$
  end while;
  $S \leftarrow S \cup \{c_i\}$
end while;
if $i = n + 1$ **then** $S \leftarrow \{\}$ **fi**

- **計算量はプログラムサイズに対してほぼ線形オーダー。**

節中の変数の出現の仕方に基づく最適化

優先度づけ Kima は節中における変数の出現の仕方によって修正案に優先度をつける。

- 変数は一対一通信に使われるのがもっともらしい。
- もっともらしい 2回 $>$ 3回以上 $\gg$ 1回もっともらしくない

検出規則 Kima は節中に1回しか現れない変数の変数名が下線 “_” で始まっていないとき、これを誤りとして検出できる。

修正案探索の際、書き換えていない節に関しては優先度づけや検出規則による再検査を行なう必要がない。

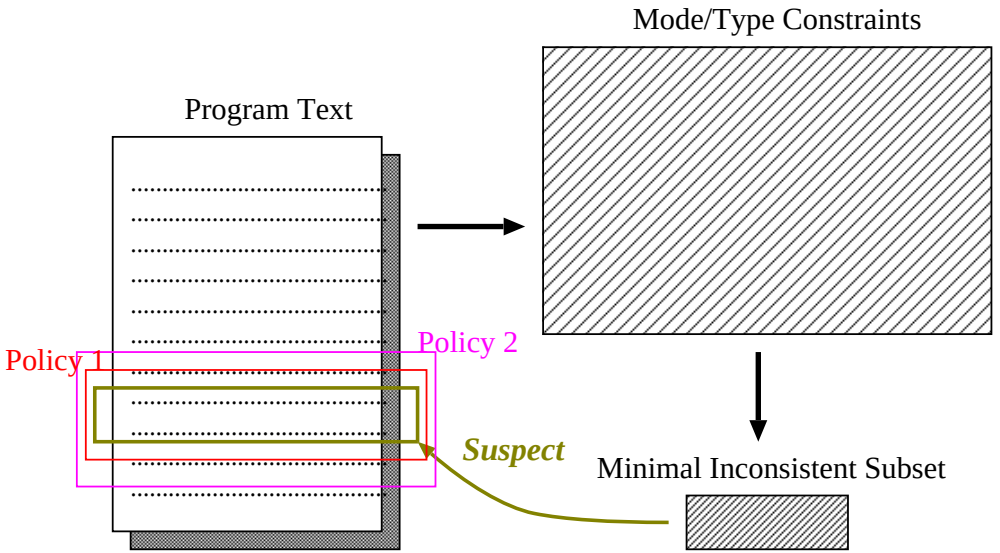
モード/型の再計算には、プログラム全体の解析が必要。

局所的な解析による最適化

プログラム全体のモード/型解析の前に、矛盾する極小部分集合によって指摘された節の周辺だけを解析する。

方針1 極小部分集合によって指摘された節を含む述語 (第一述語と呼ぶ) の述語定義節

方針2 第一述語に加えて、第一述語を呼び出している述語と第一述語が呼び出している述語の述語定義節



2種類の最適化を利用したアルゴリズム

最適化したアルゴリズム

1. モード/型制約の矛盾する極小部分集合の計算による、誤り箇所候補の特定
2. 以下の手続きの繰返しによる generate-and-test
   - (a) 特定した場所の周辺の機械的な書換えの中で優先度の高いものだけをピックアップ
   - (b') 検出規則による検査
   - (b'') モード/型の局所的な検査
   - (b) プログラム全体のモード/型の検査

実験結果 — 最適化による探索空間の縮小効果

プログラム	節の数	節中の変数出現数の平均	深さ	すべての書換え案の数	検出規則に通った書換え案の数	局所的な検査に通った書換え案の数 方針1	局所的な検査に通った書換え案の数 方針2	最終検査に通った書換え案の数
nqueen	34	10.4	1	36.8	2.55	2.05	1.76	1.43
(67行)			2	8069	199	84.8	78.2	6.48
gen_test	73	10.8	1	66.9	2.45	2.01	1.89	1.61
(183行)			2	26700	21.5	8.78	7.90	4.25
tgraph	79	17.6	1	53.2	2.59	1.78	1.74	1.39
(218行)			2	16900	8.10	4.70	4.70	3.40
graph	155	16.8	1	86.0	2.54	1.80	1.77	1.35
(390行)			2	44000	9.40	4.20	3.90	3.10

- 優先度づけと検出規則によって、書換え案を大幅に絞り込める。
- 局所的な解析によって、最終的に求まる修正案に近いところまで、書換え案をさらに絞り込める。
- 方針1と2では効果にあまり差がない。

今後の課題

- 書換え案の生成の効率化。優先度の高い書換え案だけを必要に応じて生成できるようにする。
- 現実サイズのプログラムに対する応答時間の調査。
- 入出力例を用いることで、より高度な自動修正を実現
  (入出力例はモードと型の仕様にもなる)。

(半) 自動プログラミングへの応用

入出力例を与えることによって、変数の記述を自動化

```
:- instance qsort(+[5,2,7,3],-[2,3,5,7],+[]).
qsort/3 :- X=[]    |=/2.
qsort/3 :- X=[X|X] |
  part/4,qsort/3,X=[X|X],qsort/3.
part(_,[],    S, L ) :- true | S=[],L=[].
part(A,[X|Xs],S0,L ) :- A>=X | S0=[X|S],part(A,Xs,S,L).
part(A,[X|Xs],S, L0) :- A< X | L0=[X|L],part(A,Xs,S,L).
```

⇓ 探索による自動生成

```
qsort([],    Ys0,Ys ) :- true | Ys=Ys0.
qsort([X|Xs],Ys0,Ys3) :- true |
  part(X,Xs,S,L),qsort(S,Ys0,Ys1),Ys1=[X|Ys2],qsort(L,Ys2,Ys3).
```

まとめ

- 優先度づけや検出規則、局所的な解析が修正案の探索空間の縮小に非常に効果的であることを示した。
- Kima はそれ自身 KL1 で記述されており、現在の大きさは約 4,500 行。
- Kima は以下のサイトからフリーソフトとして利用可能

  http://www.ueda.info.waseda.ac.jp/~ajiro/study-j.html