美しい日本のMLコンパイラ

開発代表者：
住井 英二郎
University of Pennsylvania

目的

短期的には…

「きれい」でわかりやすい
MLコンパイラを作る

中期的には…

ML自体の宣伝

長期的には…

プログラミング言語研究の
「技術移転」

What is ML?

- ML ≠ Mailing List
- ML ≠ Markup Language

ML = Meta Language

単純で強力なプログラミング言語

- 主に関数型＋命令型、オブジェクトも有り
  - Milnerらにより設計・実装（チューリング賞）
  - Appel, Leroyらにより発展（SML/NJ, OCaml）

どう「単純で強力」なのか？

- Computer Language Shootout (shootout.alioth.debian.org)
  - 様々な言語のマイクロベンチマーク

	1位	2位	3位	4位	5位	6位
性能	Intel C	gcc	MLton	Clean	OCaml	g++
行数	Haskell	Perl	Ruby	Nice	S-Lang	SML/NJ

- ACM ICFPプログラミングコンテスト
  - プログラミング言語は自由
  - プロも参加（Leroy, Peyton Jones, topcoder, ...）

ICFPコンテスト

年度	課題	1位	2位
98	○×ゲーム	並列C	OCaml
99	DSL最適化	OCaml	Haskell
00	レイトレーシング	OCaml	OCaml
01	HTML最適化	Haskell	Dylan
02	荷物配送ゲーム	OCaml	C
03	レーシング	C++	C++
04	アリ対戦	Haskell & C++	Haskell

私のML宣伝活動

- ACM ICFPプログラミングコンテスト
  - 2000年度優勝 (Penn)
  - 2002年度優勝（東大）
  - 2004年度主催 (Penn)
- MLプログラミング・コンパイラ授業（東大理学部情報科学科）[CPU実験]
- Bioinformaticsに使用（東大医科研）
- O'Reillyの本を共同翻訳
- 自分近傍のプログラミングはすべてML

なぜ美しいコンパイラを作るのか

MLが"Minor Language"である原因：

- 「よく知らないから使わない」
- 「仕組みがわからないから使わない」

⇓

プログラマや学生にわかりやすい
チュートリアルや実装や公開
（短いほど良い）

対象とするMLサブセット(1/2)

M, N ::=	式
c	定数
$op(M_1,...,M_n)$	算術演算
if M then N_1 else N_2	条件分岐
let $x = M$ in N	変数定義
x	変数読み出し
let rec $x(y_1,...,y_n) = M$ in N	関数定義
$M(N_1,...,N_n)$	関数呼び出し
...	

対象とするMLサブセット(2/2)

M, N ::=

...	
$(M_1,\ldots,M_n)$	組を作る
let $(x_1,\ldots,x_n) = M$ in N	組から読み出し
Array.create M N	配列を作る
$M.(N)$	配列から読み出し
$M_1.(M_2) \leftarrow M_3$	配列へ書き込み

プログラム例

```
let rec gcd m n =
   if m = 0 then
      n
   else if m <= n then
      gcd m (n - m)
   else
      gcd n (m - n)
in gcd 21600 337500
```

コンパイルの例

```
let rec gcd m n =
   if m=0 then n
   else if m<=n then gcd m (n-m)
   else gcd n (m-n)
```

```
gcd.19:
  cmp  %i2, 0
  bne  be_else.45
  nop
  mov  %i3, %i2
  retl
  nop
be_else.45:
  cmp  %i2, %i3
  bg   ble_else.47
  nop
  sub  %i3, %i2, %i3
  b    gcd.19
  nop
ble_else.47:
  sub  %i2, %i3, %o5
  mov  %i3, %i2
  mov  %o5, %i3
  b    gcd.19
  nop
```

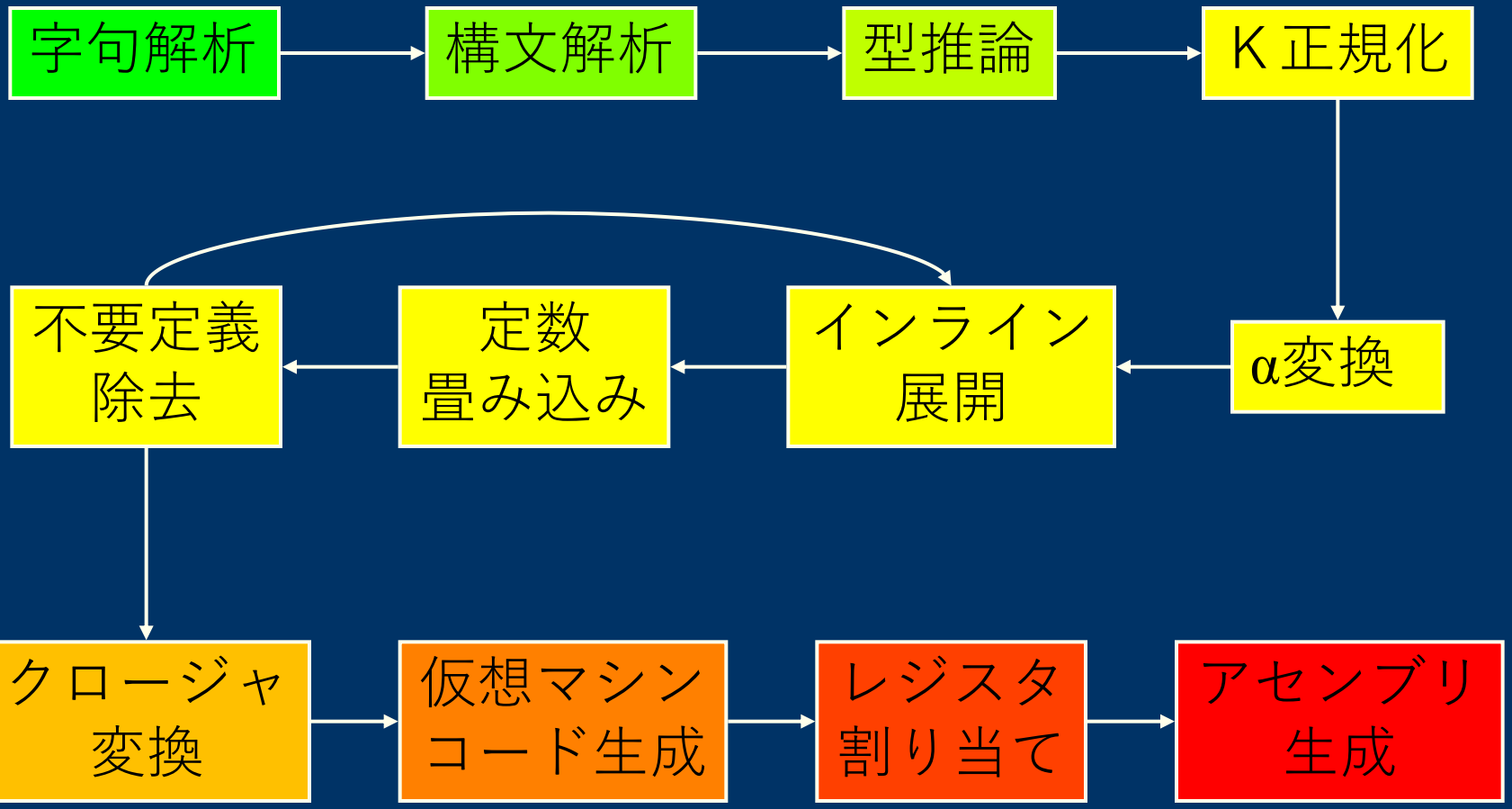
実装方針

コンパイル　＝

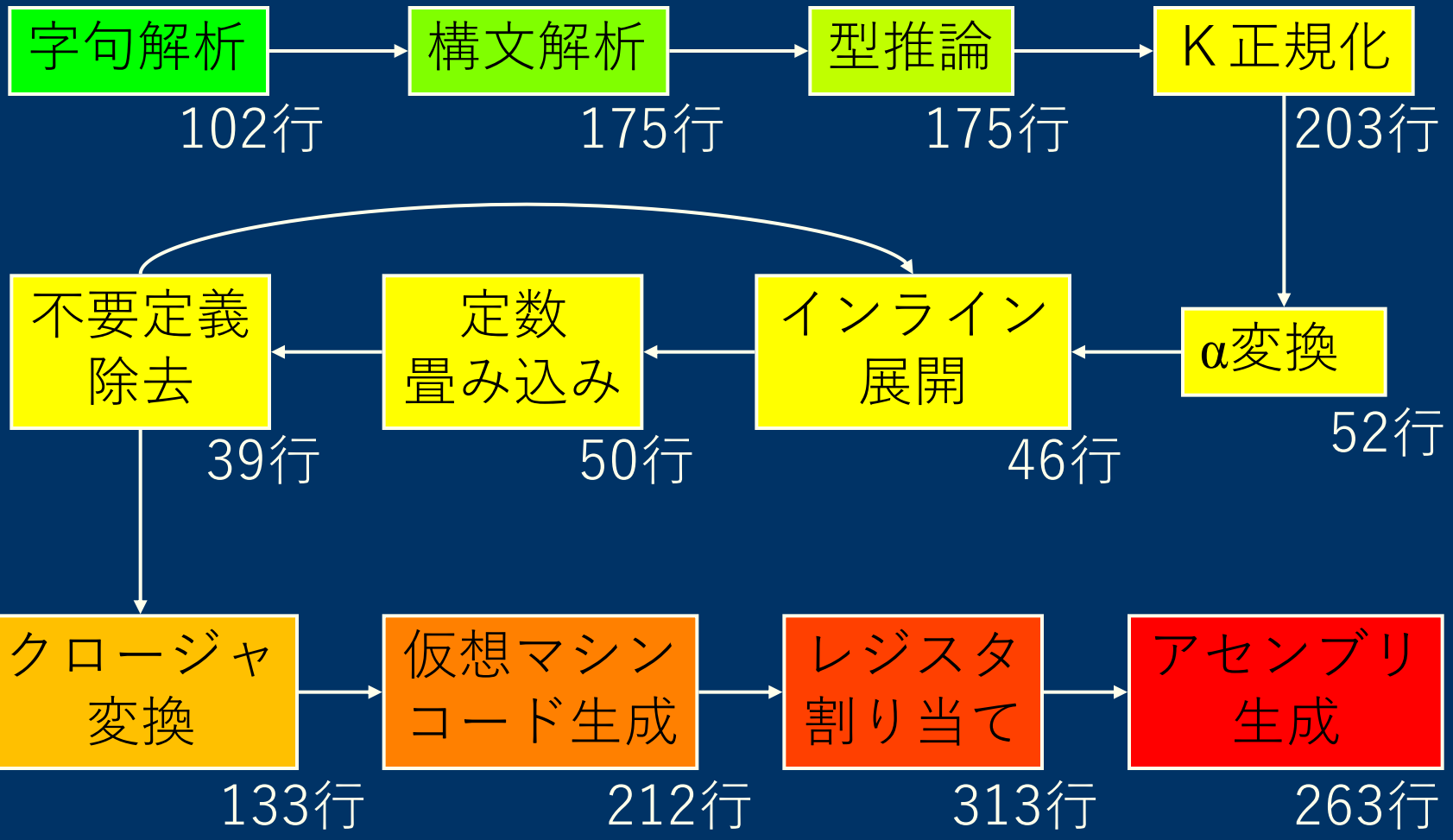
ある言語をlow-levelな言語に変換すること

適切な中間言語を設定すれば、
単純な変換の連続

コンパイラの構成


字句解析
構文解析
型推論
K 正規化
α変換
インライン
展開
定数
畳み込み
不要定義
除去
クロージャ
変換
仮想マシン
コード生成
レジスタ
割り当て
アセンブリ
生成

コンパイラの構成


字句解析
102行
構文解析
175行
型推論
175行
K 正規化
203行
α変換
52行
インライン
展開
46行
定数
畳み込み
50行
不要定義
除去
39行
クロージャ
変換
133行
仮想マシン
コード生成
212行
レジスタ
割り当て
313行
アセンブリ
生成
263行

補助モジュールも合計すると…

```
> wc -l *.ml{l,y,i,}
     100 lexer.mll
     171 parser.mly
       (中略)
    2262 total
```

2千行でMLサブセットの
コンパイラができた！

性能

典型的な関数型プログラム（Ackermann関数）

OCamlOpt	0.9秒
未踏MLコンパイラ	0.9秒
gcc	6.8秒
gcc -mflat	1.3秒

典型的な命令型プログラム（レイトレーシング）

OCamlOpt	27.2秒
未踏MLコンパイラ	14.1 秒
gcc（参考）	7.0秒

コンパイラの詳細：K正規化

```
(a + b) - (c * d)
```

のようなネストした式に対し、

```
let tmp1 = a + b in
let tmp2 = c * d in
tmp1 - tmp2
```

のように中間値をすべて明示化する

- MLとアセンブリのギャップ（の一つ）が埋まる
- 様々な最適化等が簡単になる

コンパイラの詳細：最適化等

K 正規化すれば非常に簡単

- インライン展開 (inline.ml)
- 定数畳み込み (constFold.ml)
- 不要定義削除 (elim.ml)

成果

- コンパイラは完成
- アルゴリズム・ソースコードも「きれい」
- テストプログラム25個が動作
- レイトレーシングも動作
- 東大、お茶の水大、東工大、JAISTなどで使用

今後の方向

- ドキュメントの整備
  - チュートリアル「30分でわかるOCaml」も予定
- IA-32アセンブリを生成
  - SPARCよりややこしい（2オペランドなど）
- サブセットの拡張
  - データ型、モジュール、Garbage Collectionなど

蛇足：プログラミング言語研究の「技術移転」について

- Garbage Collection
  - 研究：Lisp（60年代）
  - 普及：Java（90年代〜）
- Parametric Polymorphism
  - 研究：ML（70年代）
  - 普及：C++のSTL（90年代〜）
    C#やJavaのGenerics（つい最近）

時間がかかりすぎ！
もっと短縮できないか？