1 Keywords

- tail call と続継と CPS と空間 complexity(PTR)
- 遅延分割の方法
- 共通中間言語（GC 付）

2 Tail Call

- tail call のときに，明示的スタックを伸ばさない.
- tail call 用の呼出し構文 (jump g a)

```
(define a (array int 100))
(jump g a)
```

とした場合に，a の先は callee で使われるのでつぶしてはいけない.

3 参照

呼出し先に渡す以外の渡し方をしてはいけないようなポインタ型 → C ではポインタを return したいことがよくあるので問題.

呼出し先に渡す or 指定回数以内での return をしてはいけないような”ポインタ”を導入するとよい（要するに escape してはいけない）.

```
(def p ((var 2) int))
```

の場合，p は 2 回まで return してよい

- キャストは数字が小さくなる方向にのみ可能
- (return する変数の var 数-1) ≥ (関数の返り値の型の var 数)
- (関数呼出しの実引数の var 数+1) ≥ （関数の引数の型の var 数)
- 通常のポインタは var 数無限大

Pascal の”var”，C++の”参照”との関係？

4 明示的スタックへの保存

”保存”時の”変換”は，C スタック全体（のうち未変換部分）が対象.

C スタックを巻き戻しながら，明示的スタック領域の base と逆側からフレームを保存していき，最後に reverse して明示的スタックに積むと，通常実行時の明示的スタックポインタの増減をサボれる.

5　C スタックの再構築

”再開”時は，C スタック全体を再構築せず，”トップ”のみ再開．このとき，引数や返り値の授受の”一般化”を仮定．そのための”ラッパー”が必要（明示的スタック上で授受）．

6　Proper tail recursion

C のスタックが溢れそうになったら，”保存”，”再開”を実行すると，Tail call をしたフレームは潰れる．