タイトル

名前　(指導教員：浅井 健一)

1　はじめに

卒論の 2 ページの予稿のサンプルです。最初に背景から概要を書きます。

卒論の予稿は短いので書きませんが、PPL の論文など長い論文では以下のような全体の構成を書きます。これが、節の番号を参照する例になっています。

本論文の構成は以下のようになっている。まず、最初に 2 節でラムダ式の書き方について説明する。3 節でいろいろな箇条書きの書き方について説明し、4 節ではコードの書き方の例を紹介する。5 節で参考文献について触れる。6 節で定理の書き方を、7 節で証明木の書き方を示す。9 節でまとめる。

2　ラムダ式

macro.tex に書いてあるような定義をあらかじめしておいて、それを使うのが良いです。

$$\begin{array}{rrcl} \text{型} & t & := & \mathsf{int} \mid t \to t \\ \text{項} & e & := & x \mid \lambda x.\, e \mid e\, e \mid \mathcal{S}k.\, e \mid \langle e \rangle \end{array}$$

3　箇条書きの書き方など

箇条書き

- ひとつめ
- ふたつめ
- みっつめ

番号付き

1. ひとつめ
2. ふたつめ
3. みっつめ

見出し付き

継続　その後の計算をさす。

限定継続　継続のうち、その範囲が限定されているもの。

4　コード

コードを直接、書くには verbatim 環境が簡単です。

```
let rec fac n =
  if n = 0 then 1 else n * fac (n - 1)
```

verbatim 環境内で tex のコマンドを使いたいときは、alltt を usepackage して使います。上のコードは、どうも前後の文と間がきつすぎると思うときは、quote 環境に入れるというのはひとつの手です。

```
let rec fib n =
  if n < 2
  then n
  else fib (n - 1) + fib (n - 2)
```

これらの環境はタイプライタフォントなので、横幅をとりすぎる傾向にあります。なれてきたら、よりきれいにコードを書く方法を習得するのが良いかも知れません。

5　参考文献

bibtex を使うのが良いでしょう。paper.bib に型デバッガ [6] 関係の文献と限定継続 [1, 2] 関係の文献を入れておきました。また、本の例としてアルゴリズミックデバッギング [4] も入れました。

6　定理

定義 1 (CPS 変換 [3]) 項 e の CPS 変換 $[\![e]\!]$ は以下のように定義される。(中身は省略。)

命題 2 e に型がつくなら、その部分式にも型がつく。

補題 3 (代入補題 [5]) $x : t_1 \vdash e : t$ かつ $\vdash v : t_1$ なら $\vdash e[v/x] : t$ が成り立つ。

定理 4 e に型がついたら、e の実行中に型エラーは起きない。

7　証明木

証明木の例です。judgement もマクロとして定義するのが良いです。

$$\frac{\Gamma(x) = t}{\Gamma \vdash x : t}\ (\mathsf{TVar}) \qquad \frac{\Gamma, x : t_1 \vdash e : t_2}{\Gamma \vdash \lambda x.\, e : t_1 \to t_2}\ (\mathsf{TLam})$$

$$\frac{\Gamma \vdash e_1 : t_2 \to t_1 \quad \Gamma \vdash e_2 : t_2}{\Gamma \vdash e_1\, e_2 : t_1}\ (\mathsf{TApp})$$

8　長さ稼ぎ

あいうえおかきくけこさしすせそたちつてとなにぬねの

はひふへほまみむめもやゆよらりるれろわをん
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
0
1
2
3
4
5
6
7
8
9

9　まとめ

最後に、まとめと今後の課題などを書きます。このサンプルは 1 ページで終わっていますが、必ず 2 ページを埋めます。（2 ページはすぐ埋まります。むしろすぐ足りなくなります。紙面が足りない場合、下のように参考文献は多少、小さくしても構いません。）

参考文献

[1] O. Danvy and A. Filinski. A Functional Abstraction of Typed Contexts. Technical Report 89/12, DIKU, University of Copenhagen, July 1989.

[2] O. Danvy and A. Filinski. Abstracting Control. In *Proc. 1990 ACM Conference on Lisp and Functional Programming*, pp. 151–160, 1990.

[3] G. D. Plotkin. Call-by-Name, Call-by-Value, and the λ-Calculus. *Theoretical Computer Science*, Vol. 1, pp. 125–159, 1975.

[4] E. Y. Shapiro. *Algorithmic Program Debugging*. Cambridge: MIT Press, 1983.

[5] A. K. Wright and M. Felleisen. A syntactic approach to type soundness. *Inf. Comput.*, Vol. 115, No. 1, pp. 38–94, 1994.

[6] 対馬かなえ, 浅井健一. コンパイラの型推論を利用した型デバッグの手法の提案. コンピュータソフトウェア, Vol. 30, No. 1, pp. 180–186, 2013.