prosperを使おう (2)
—スタイルファイル作成—

池田 大輔* 渡部 善隆† 藤野 清次‡

1 はじめに

prosper は Frédéric Goualard によって作られた LaTeX のクラスファイルで、プレゼンテーションの資料作成に使います。前回は簡単な prosper の使い方を紹介しました [2]。今回は、独自のスタイルファイルの作成方法を説明します。また、効果的なスタイルファイル、特に背景を作成するために、前回説明できなかった PSTricks の機能をより詳しく紹介します。

図 1 と図 2 は、新たに作ったスタイルファイルを利用したスライドです[1]。スタイルファイルの作り方は 3 節を、中身の記述については 5 節を参照してください。

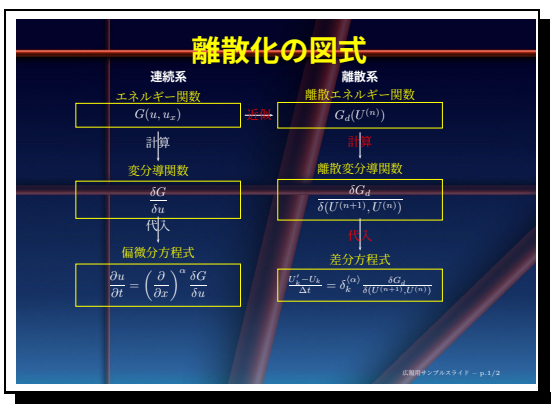

図 1: サンプルスライド (1 枚目)

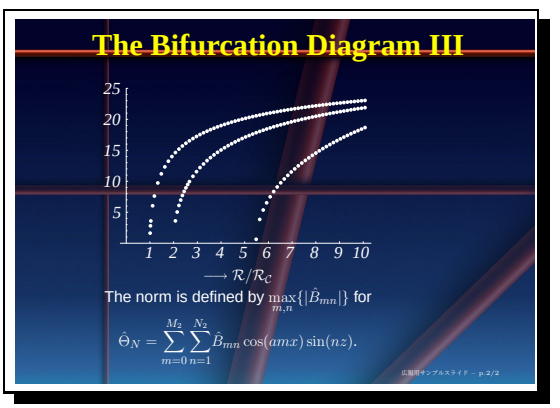

図 2: サンプルスライド (2 枚目)

前回の記事で、日本語を含んだ資料の場合は Ghostscript 付属の ps2pdf コマンドではきれいな表示ができないと書きました。このため、UNIX 内で必要なコマンドが完結せず、Windows か Machintosh 環境と Acrobat Distiller が必須でした。しかし、gs-cjk プロジェクト [10] により Ghostscript の機能拡張がなされ、gs や ps2pdf からスケーラブルなフォントである CID フォントや TrueType フォントを利用できるようになりました。本稿では、新しい Ghostscript のイン

*情報基盤センター研究部 mailto:daisuke@cc.kyushu-u.ac.jp
†情報基盤センター研究部 mailto:watanabe@cc.kyushu-u.ac.jp
‡情報基盤センター研究部 mailto:fujino@cc.kyushu-u.ac.jp

[1] このサンプルは大阪大学サイバーメディアセンター降旗先生から頂いた LaTeX ソースを修正したものです。このスライドのソースは 5 節にあります。

ストール方法と、これらのフォントを利用する方法を説明します。これで、日本語を含んだプレゼンテーション資料の場合でも、ほぼ UNIX 環境内のコマンドのみで満足のゆく資料作成ができるようになります。

1.1 必要な環境

prosper は LaTeX のクラスファイルなので、prosper を使うには LaTeX が使える必要があります。pLaTeX や jLaTeX でも構いません。

prosper はいくつかのクラスファイルやスタイルファイルを内部で利用します。そのため、あらかじめ seminar, PSTricks, hyperref (6.69 より新しいバージョン) の各パッケージがインストールされている必要があります。これらのパッケージは teTeX [8] をインストールすると、すべてインストールされます[2]。teTeX は Thomas Esser がメンテナンスしている TeX のディストリビューションの一つです。

prosper を利用するときは最終的に PostScript 形式か PDF 形式にしてからプレゼンテーションを行ないます。dvi ファイルから PostScript にするために dvips (5.85 以降) が必要です。

PostScript ファイルから PDF ファイルに変換するためには、Adobe 社の Acrobat Distiller を用いるか、Ghostscript に付属している ps2pdf コマンドを使う必要があります。

ps2pdf を用いる場合は、Ghostscript のバージョン 6.0 以降がお勧めです。これより前のバージョンでは、プロジェクタに投影する場合フォントが綺麗に表示できません。また、日本語を使う場合は、上述の gs-cjk プロジェクトの成果を含んだ 6.53 以降か 7.05 以降を使いましょう。バージョン 7.05 のインストール方法は 4.1 節を参照してください。

本稿で用いる prosper のバージョンは 1.00.4 です。確認は Linux (Vine 2.5), FreeBSD, Windows 2000 上で行ないました。Linux における各コマンドのバージョンは以下のとおりです。

```
 % platex
This is pTeX, Version p3.0.1, based on TeX, Version 3.14159 (EUC) (Web2C 7.3.1)
 % dvips
This is dvipsk 5.86 p1.5e Copyright 1996-2001 ASCII Corp.(www-ptex@ascii.co.jp)
based on dvipsk 5.86 Copyright 1999 Radical Eye Software (www.radicaleye.com)
 % gs -v
GNU Ghostscript 7.05 (2002-04-22)
```

これらのコマンドのうち platex と dvips は RPM パッケージでインストールしました。インストールしたパッケージは

- tetex-1.0.7-0vl14
- tetex-extra-1.0.7-0vl14
- tetex-macros-1.1-0vl1

です。Ghostscript はソースからコンパイルしてインストールしました。インストールの仕方は 4.1 節を参照してください。バージョン 7.05 以降の RPM パッケージもあるようですので、これを利用してもよいでしょう。

[2] pTeX はバージョン 2.1.9 から teTeX ベースになりましたので、pTeX をインストールしても、これらのパッケージはインストールされます。角藤氏による Windows 版 pTeX のバイナリパッケージでも同様です。

FreeBSD における各コマンドのバージョンは以下のとおりです。

```
 % platex
This is pTeX, Version p2.1.11, based on TeX, Version 3.14159 (EUC) (Web2C 7.3.1)
 % dvips
This is dvipsk 5.86 p1.5g Copyright 1996-2002 ASCII Corp.(www-ptex@ascii.co.jp)
based on dvipsk 5.86 Copyright 1999 Radical Eye Software (www.radicaleye.com)
 % gs -v
GNU Ghostscript 7.05 (2002-04-22)
```

これらのコマンドはすべて ports からインストールしました。Ghostscript は ghostscript-gnu-7.05_1 をインストールした上で、4.2 節の作業を行い日本語フォントを利用できるようにしました[3]。

また、Windows2000 上の pTeX (Version p2.1.11, based on TeX, Version 3.14159 (SJIS) (Web2C 7.3.3)) と Ghostscript 7.03 でも動作を確認しています。Windows 上の Ghostscript でも、バージョン 7.05 以降は CID フォントや TrueType フォントが使えます。利用するための設定は、基本的に 4.1 節で説明する UNIX 上のものと同じです。

上記のコマンドやスタイルファイル等は、研究用システムのライブラリサーバ wisdom 上で使うことができます。このうち platex などのコマンドは/usr/local/bin にインストールしてあります。

```
 % platex
This is pTeX, Version p3.0.1, based on TeX, Version 3.14159 (EUC) (Web2C 7.3.1)
 % dvips
This is dvipsk 5.78 p1.4c Copyright 1996-99 ASCII Corp.(www-ptex@ascii.co.jp)
%%dvipsk 5.78 Copyright 1998 Radical Eye Software (www.radicaleye.com)
 % gs -v
GNU Ghostscript 7.05 (2002-04-22)
```

また、prosper そのものは/usr/local/share/texmf/tex/misc/prosper にインストールしてあります。通常は、これらのコマンドやクラスファイルは設定の変更なしに利用できると思います。

2 PSTricks

PSTricks パッケージは、T. Van Zandt による PostScript の描画能力を LaTeX から使うためのパッケージです。前回の記事 [2] でも簡単な使い方を紹介しました。PSTricks を利用すれば、スライド中に図形を描画することもできますし、スライドの背景を描画するためにも使うことができます。スライドの背景を書く場合には、単に図形を線で書くだけでなく、塗りつぶし、特にグラデーションによる塗りつぶしをよく使うと思います。特定の図形の繰り返し描画もよく利用されます。今回は、主にスライド背景を描画する時に便利な機能を中心に説明します。

[3]Ghostscript として ja-ghostscript-gnu-jpnfont-7.05 をインストールすれば、4 節で説明する東風フォントや CMap ファイルのインストールと東風フォントを使うための設定まで行なってくれます。

2.1 図形描画

まず、単純な線を書きましょう。線は\psline で書きます。一般形は

$$\backslash\mathtt{psline}[\mathit{options}]\{\mathit{arrows}\}(x_1,y_1)(x_2,y_2)\ldots(x_n,y_n)$$

で、(x_1,y_1) から各点 (x_i,y_i) $(2 \leq i \leq n)$ を通り、(x_n,y_n) まで折れ線を引きます。

{*arrows*}を省略すると通常の線を引きますが、表 1 のような指定をすることで矢印にすることができます。左右対象である必要はありません。

表 1: 矢印の種類

値	出力例	値	出力例
-		<->	
>-<		<<->>	
>>-<<		|-|	
|*-|*		[-]	
(-)		o-o	
-		oo-oo	
-		c-c	
cc-cc		C-C	

線の太さ (linewidth でデフォルトは 0.8pt) や線種 (linestyle でデフォルトは solid)、線の色 (linecolor でデフォルトは black) は、省略可能なオプション引数 [*options*] で変えます。例えば、線の太さと線種を変えるときは、

```
\psline[linestyle=dotted, linewidth=3pt](0, 0)(2, 2)
```

とします。linestyle として、他に none と dashed が使えます。

\psset コマンドを使えば、線種や色などを恒久的に変更することができます。例えば、さきほどの例は、

```
\psset{linestyle=dotted, linewidth=3pt}
\psline(0, 0)(2, 2)
```

とすることもできます。ただし、この場合はこれ以降の描画コマンドにおいて、常に linestyle は dotted で、linewidth は 3pt になります。

2.2 \rput

\rput は前回の記事 [2] で簡単に紹介しましたが、直接描画するコマンドではなく、テキストや図を指定した位置に配置するコマンドです。このとき、配置するものを回転させることもできます。\rput の一般形は

\rput[*ref*]{*rot*}(*x*,*y*){配置したいもの}

です。(*x*,*y*) は原点の指定で、通常は{配置したいもの}の中央をどこに置くかを意味します。[*ref*] と{*rot*}は省略できます。{*rot*}は回転の角度を指定します。

原点に合わせる場所を変更するには [*ref*] を指定します。水平方向の l (left), c (center), r (right) と、垂直方向の t (top), b (bottom), B (baseline) が指定可能です。例えば、{配置したいもの}の左上を原点に合わせるためには、オプション [lt] を指定します。

図 3 に [*ref*] を変えた時の位置の違いを示します。左側は水平方向の違いを、右側は垂直

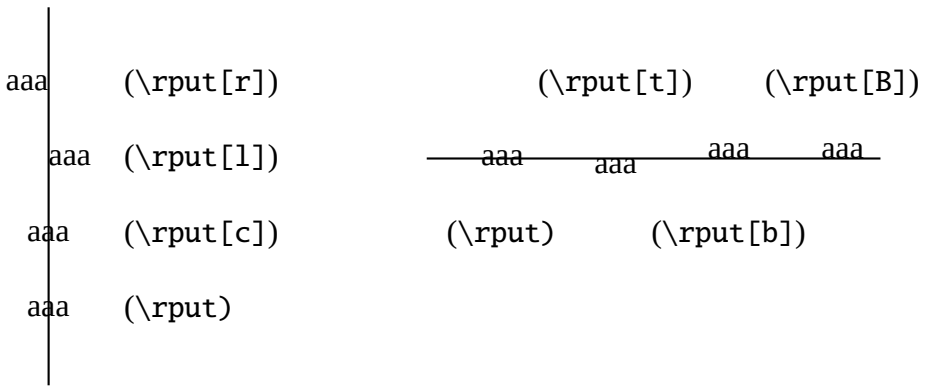

図 3: ref の指定による位置の違い

方向の違いを示しています。

2.3 カラー

PSTricks では、あらかじめ red, green, blue, cyan, magenta, yellow の各色と、グレースケールの色として black, darkgray, gray, lightgray, white が定義されています。

新しい色を定義するには

\newrgbcolor{色の名前}{n_1␣n_2␣n_3}

とします。n_1, n_2, n_3 は RGB カラーモデルにおける赤、緑、青の割合を 0〜1 までの実数で表わしたもので、空白で区切ります。例えば、\newrgbcolor{mycolor}{0.28 0.24 0.68}などとします。

新しいグレースケールの色を定義するには\newgray{色の名前}{*n*}とします。*n* は 0〜1 までの実数です。

定義した色を利用する場合は、PSTricks のコマンドの色を指定するところに定義した色の名前を書きます。例えば、線の色をさきほど定義した mycolor を使って線を書く場合は、

```
\psline[linecolor=mycolor](0, 0)(2, 2)
```

などとします。

テキストの色を変更するには、“{\mycolor ここの色が変わります}” のように使います。この場合、色の名前の前に\ (バックスラッシュ) が必要です。

✔ prosper 付属のスタイルのうち darkblue, alienglow, autumn, defaults 以外のスタイルでは、itemize 環境や itemstep 環境の中の\item コマンドの直後で上のように色指定をしても有効になりません。これら 4 つのスタイルでは\myitem を宣言していますので、もともと prosper で定義されている行頭文字の定義がおかしいようです。とりあえず、色の指定の前に空の\mbox を置いて “\mbox{}{\mycolor ここの色が変わります}” とすると色が変わります。

2.4 グラデーション

グラデーションを使うには、プリアンブルで\usepackage{pst-grad} を宣言する必要があります[4]。図 4 はパラメータを変えて描画したグラデーションの例で、以下のようにして描画

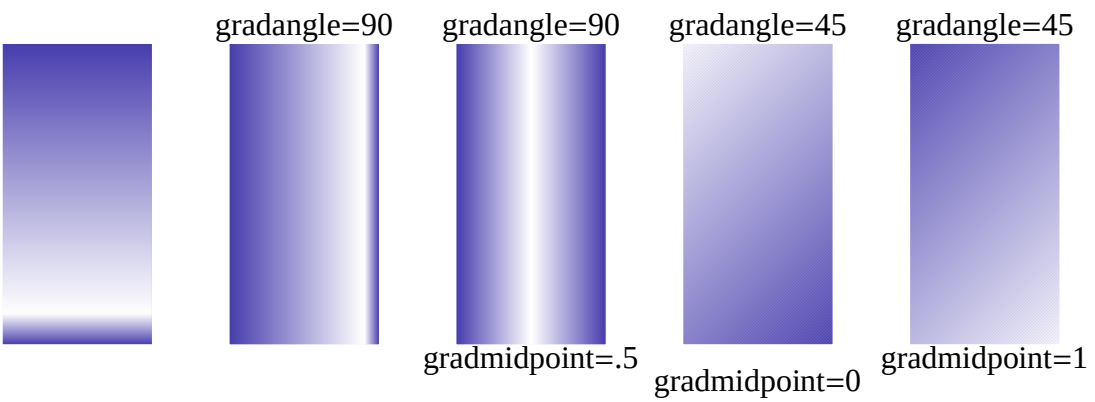

図 4: グラデーションの使い方

しています。

```
\begin{pspicture}(-4, 0)(\textwidth, 4)
  \psset{linestyle=none,fillstyle=gradient,%
    gradbegin=mycolor,gradend=white}
  \psframe[gradangle=0](2,4)
  \psframe[gradangle=90](3,0)(5, 4)
  \psframe[gradmidpoint=.5,gradangle=90](6,0)(8, 4)
  \psframe[gradmidpoint=0, gradangle=45](9,0)(11, 4)
  \psframe[gradmidpoint=1, gradangle=45](12,0)(14, 4)
\end{pspicture}
```

グラデーションは、塗りつぶしの一種なので fillstyle=gradient とする必要があります。その上で、どの色からどの色へ変化させるかを gradbegin と gradend に指定します。自分で定義した色も指定可能です。図 4 は自分で定義した mycolor から白へのグラデーションです。

色の変化の方向は gradangle によって指定します。指定を行なわない場合は、図 4 の一番左のように、縦方向へ色が変化します。次の例では gradangle=90 することにより、水平方向への変化にしています。gradangle=45 とすると斜め方向への変化になります。

[4]使っているオプションによっては pst-grad パッケージがあらかじめ読みこんである場合もあります。その場合はこの宣言は不要ですが、宣言しても害はありません。

色の変化が完了する地点を gradmidpoint で定義します。値は 0 から 1 までの実数値で、デフォルトでは .9 となっています。したがって、図 4 の左 2 つの例は、上から 9 割の地点と左から 9 割の地点で、白への変化が完了しています。真ん中の例では gradmidpoint=.5 として中央で白になるようにしています。右 2 つの例では gradmidpoint=0 と gradmidpoint=1 とすることにより、一番上または下で白になるようにしています。

2.5 描画の繰り返し

\multirput を使うと、ある描画処理を繰り返すことができます。\multirput の一般形は、

\multirput[*ref*]{角度}$(x_1, y_1)(x_2, y_2)${個数}{書きたいもの}

となります。

座標は、最初に書く位置と次に描画する位置までの距離です。つまり (x_1, y_1), $(x_1 + x_2, y_1 + y_2)$, $(x_1 + 2x_2, y_1 + 2y_2)$, ... の位置に描画されます。次に何個書くか指定します。角度は書きたいものを個々に回転させるためのもので、省略可能です。[*ref*] は rput のそれと同じです。

例えば、以下のように入力すると、図 5 のように描画されます。

```
\multirput(0,0)(2, .25){8}{+}
```

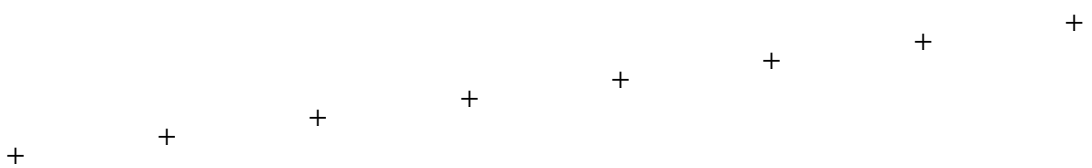

図 5: \multirput による繰り返し描画の例

3 独自のスタイルを作ろう

この節では、スライドの背景などを変更し、自分独自のスタイルを作る方法を説明します。

基本的なスタイルの作り方は、各スライドにおけるタイトルと本文の色やフォント、位置などを決めたり、背景となる部分を自分で描画します。ここで、タイトルとは slide 環境の引数となるもので、図 8 においては “The quest for π” です。

背景を描画する方法は、LaTeX からできる事であれば何でも構いません。例えば、背景となる絵を別のソフトウェアで作成し、これを graphics パッケージか graphicx パッケージの \includegraphics で貼りつけることもできます。picture 環境や 2 節で説明した PSTricks も利用できます。

3.1 長さと座標

prosper に付属しているスタイルを見ると、位置や長さを絶対的な数値で指定している箇所が多数あります。指定する単位には、通常の LaTeX で使えるものを使います。

あらかじめ定義してある長さの変数には、スライドの横幅を格納する\slideWidth があります。この値は、3.2 節で説明する\NewSlideStyle の最初のオプション引数で与えた長さになります。オプションですので省略可能で、省略された場合は 11cm になります。

座標を指定して描画する場合は x 軸は現在の描画位置より右方向へ、y 軸は上方向へ伸びています。つまり、図 6 のように本文[5]に何も書いていない状態では左上が原点 (0, 0) となります。

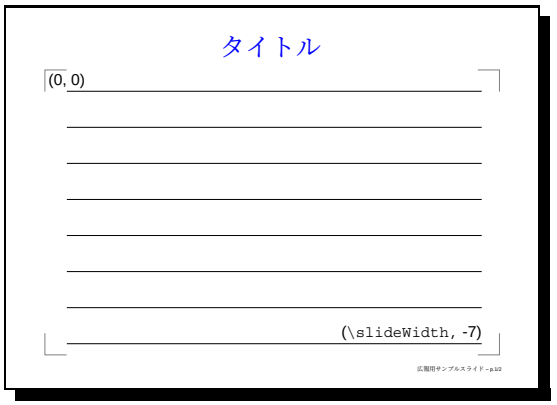

図 6: TEX の座標系

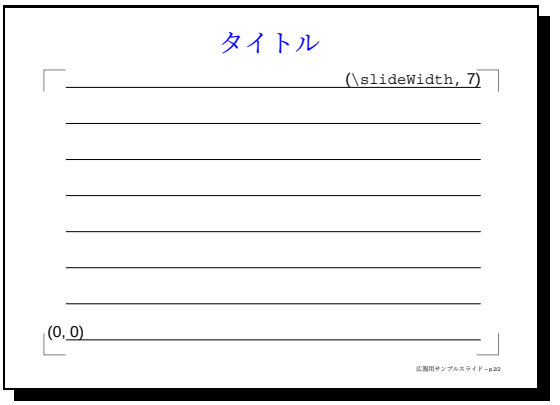

図 7: pspicture の座標系

この例では右下が (\slideWidth, -7) です。

一方、図 7 は PSTricks の pspicture 環境を利用しています。pspicture 環境は以下のようにして使います。

\begin{pspicture}[(x_0, y_0)](x_1, y_1)
描画コマンド
\end{pspicture}

(x_0, y_0) と (x_1, y_1) はどこからどこまでを描画領域として使うかを指定します。(x_0, y_0) は省略可能で、省略した場合は $(0, 0)$ が用いられます。pspicture 環境の内部では左下が原点になります。図 7 では、右上が (\slideWidth, 7) です。

スタイルファイルを作る時にも、タイトルの位置や背景の図形の位置などを指定する必要があります。この時は、pspicture を指定したのと同様に左下が原点になります。

3.2 既存スタイルの例

最初に、既存のスタイルファイルを例にとり、prosper におけるスタイルファイルの作り方を簡単に説明します。最も簡単な default スタイル (ファイル名 PPRdefault.sty) を見ていきましょう。このスタイルは、スタイルオプションを指定しなかった時に使用されるスタイルであり、図 8 のような見栄えです。タイトルが青で中央寄せにしてあり、本文部分の四隅には枠があります。

最初のコメントの後に、フォーマットファイルの指定やパッケージの説明などが続きます。また、スタイルファイルのバージョン情報などを出力しています。次に、必要なパッケージを \RequirePackage で読み込んでいます。

[5]タイトルと本文は独立の座標系を持ちます。

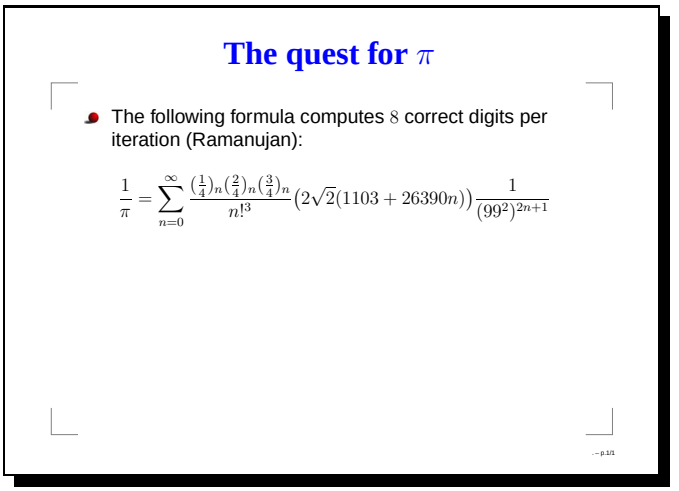

図 8: default スタイルを用いた例

```
\RequirePackage{amssymb}
```

default スタイルでは、この他に shemhelv パッケージが読みこまれています。これは prosper が内部で利用している seminar パッケージに付属するオプションスタイルファイルで、PostScript フォントの Helvetica を使うときに指定します。

次に PSTricks のコマンドを用いて色を定義しています。

```
\newgray{grayb}{.5}
```

ここでは、グレースケールの色しか定義していません。

タイトルと本文のフォントと色は、\FontTitle と\FontText コマンドで指定します。

```
\FontTitle{%
  \usefont{T1}{ptm}{b}{n}\fontsize{20.74pt}{18pt}\selectfont\blue}{%
  \usefont{T1}{ptm}{b}{n}\fontsize{20.74pt}{18pt}\selectfont\blue}
\FontText{%
  \black\usefont{T1}{phv}{m}{n}\fontsize{12.4pt}{12pt}\selectfont}{%
  \black\usefont{T1}{phv}{m}{n}\fontsize{12.4pt}{12pt}\selectfont}
```

どちらにおいても、カラーの場合とモノクロの場合を別々に指定します[6]。フォントのサイズはタイトルを大きく目立つようにしています。フォントは、タイトルには ptm (PostScript Times) のボールドシェープを、本文には phv (PostScript Helvetica) を利用しています。

次にスライドタイトルの位置を決めるために、以下のように\slidetitle というコマンドを\newcommand で定義します。このコマンドが prosper.cls から呼びだされます。

```
\newcommand{\slidetitle}[1]{%
  \rput[c](5.25,4.4){\fontTitle{#1}}
}
```

[6] この例では、どちらも同じフォントと色を指定していますが。

\fontTitle コマンドは、上で定義したタイトル部分用のフォントと色で、引数で与えられたテキストを描画します。\rput は指定した座標にテキストや図などを配置します。詳しくは2.2 節を参照してください。(0, 0) を指定するとスライドの左端の中央に配置されます。

次にロゴの位置を決めます。

```
\LogoPosition{-1,-1.1}
```

ロゴは\Logo(x,y){mylogo}か\Logo{mylogo}で指定しますが、位置の指定のない場合にここで指定した位置が用いられます。

次に背景を定義します。まず「フレーム」を定義し、このフレームを\NewSlideStyle に渡します。フレームが各スライドの背景となります。ここで背景となる絵を読み込んだり、この例のように PSTricks で図形を描画したり、グラディエーションで色を変化させたりします。

```
\newcommand{\BasicFrame}[1]{%
\psline[linewidth=.5pt,linecolor=grayb](-1,0)(-1,-0.6)(-0.4,-0.6)
\psline[linewidth=.5pt,linecolor=grayb](11,-0.6)(11.6,-0.6)(11.6,0)
\psline[linewidth=.5pt,linecolor=grayb](-1,6.7)(-1,7.3)(-0.4,7.3)
\psline[linewidth=.5pt,linecolor=grayb](11.6,6.7)(11.6,7.3)(11,7.3)
  \PutLogo % Mandatory
 {#1}}
\NewSlideStyle[115mm]{t}{5.3,3.2}{BasicFrame}
\PDFCroppingBox{10 40 594 800}
```

このフレームは BasicFrame という名前です。名前は好きにつけて構いません。このフレームでは四隅の枠を\psline コマンドで描画しています。フレームを定義する場合の座標系は、スライドの左下が原点になります。

\NewSlideStyle の一般形は

\NewSlideStyle[*width*]{ref}{pos}{frame}

となります。frame に、すでに定義したフレームを渡します。これ以外のパラメータは、背景ではなく “本文” 部分に対するもので、[*width*](省略可) は本文の長さを、{ref}と{pos}で本文のどの位置を背景のどこに置くかを指定します。{ref}の指定の仕方は、2.2 節で説明した\rput を参照してください。例えば、本文を少し広めにしてすこし左に寄せたい場合は、

```
\NewSlideStyle[120mm]{lt}{-1.2,3}{BasicFrame}
```

とします。本文の左の位置で座標を指定するために、デフォルトの t ではなく lt としています。このコマンドにおける原点は、スライドの左端の中央です。

\PDFCroppingBox{10 40 594 800}は、最終的な PDF ファイルにおいてスライドの一部を抜き出して表示させるためのものです。図 9 と図 10[7]は違う\PDFCroppingBox で作成した PDF ファイルを、Acrobat Reader で全画面表示した時の様子を取り込んだものです。このコマンド

[7]これらの図において、一番外の枠と影の部分は\shadowbox により描画しています。

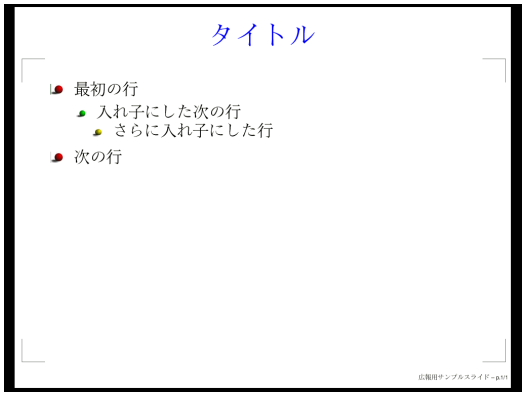

図 9: \PDFCroppingBox の引数を{10 40 594 800}とした場合

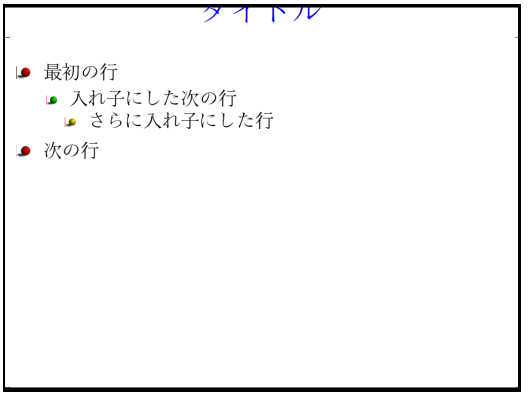

図 10: \PDFCroppingBox の引数を{50 80 554 760}とした場合

は PostScript ファイルの段階では影響はありませんので、これらの PDF ファイルの元になった PostScript ファイル (図 11) は同じように描画されます。Acrobat Reader で見ると、図 9 のほうが広い範囲を描画していることが分かります。図 10 では、上部のタイトルがきれてしまい、また、左側の余白も狭くなっていることが分かります。

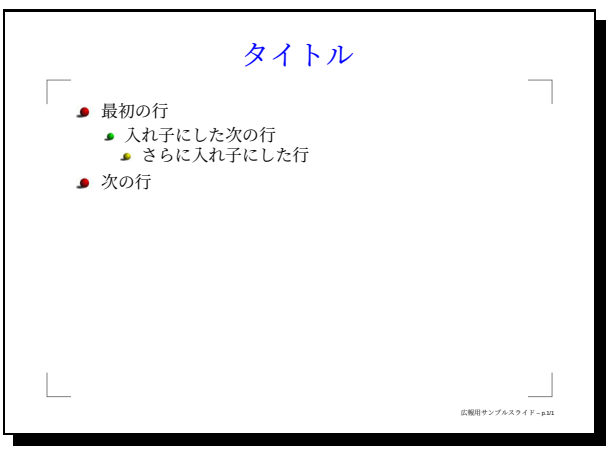

図 11: 図 9 と図 10 の元になる PostScript ファイル

スタイルファイルの最後は、itemize 環境や itemstep 環境の行頭文字の指定です。入れ子の深さを\myitem の最初の引数の数字で指定します。ここの数字は 3 までです。

```
\myitem{1}{\includegraphics[width=.4cm]{red-bullet-on-white.ps}}
\myitem{2}{\includegraphics[width=.3cm]{green-bullet-on-white.ps}}
\myitem{3}{\includegraphics[width=.3cm]{yellow-bullet-on-white.ps}}
```

ここでは prosper に付属している PostScript ファイルの絵を指定していますが、普通の文字でも構いません。

他のスタイルファイルでは、これら以外に\ColorFoot{色}コマンドを用いてフッターの色を指定しているものもあります。このコマンドは\slideCaption{キャプション}で指定したテ

キストの色を変更します。

3.3 独自のスタイルを作ろう

スタイルファイルの名前は PPR で始まり .sty で終ります。例えば azure スタイルの場合は、ファイル名は PPRazure.sty です。

この節では 4 つのスタイルを作ります (図 12 から図 15 まで)[8]。図 13 を除き、背景 (の一部) に別に作っておいた画像を使っています。図 13 は画像を使うかわりに、pTEX に付属していた ascgrp フォントと PSTricks を利用してノート用の紙のような背景を描画しています。図 12 の画像は、PowerPoint の一部に使われていた BMP ファイルを EPS に変換したものです。図 14 と図 15 は、MagicPoint [11] の背景用に作られた JPEG 画像を EPS ファイルに変換して使っています。これらの JPEG 画像は MagicPoint の背景用の画像を提供している “The MagicPoint Gallery” [12] において、GNU General Public License に基づき配布されています。

dadstie スタイル (図 12)

このスタイルは default スタイルのフレームの定義と行頭文字を変更しています。フレームは、画像ファイル DadsTie.eps があるとした時、以下のように定義します。

```
\newcommand{\BasicFrame}[1]{%
\rput[lb](-1.6,-2){\includegraphics[height=11cm]{DadsTie}}
  \PutLogo % Mandatory
 {#1}}
```

\includegraphics コマンドは画像の取り込みを行いますが、これが\rput コマンドの中で使われていることに注意してください。\rput がないと画像ファイルの分のスペースを消費していまい、あとから配置する本文部分の位置がずれてしまいます。

行頭文字は青い色を使うようにして、以下のように定義しました。

```
\myitem{1}{\blue ■}
\myitem{2}{\blue ー}
\myitem{3}{\blue ・}
```

notebook スタイル (図 13)

特別なフォントを使うために ascmac パッケージを読み込んでいます。

```
\RequirePackage{ascmac}
\newfont{\anote}{ascgrp scaled\magstep1}
```

[8] これらのスタイルファイルは PPRdefault.sty をコピーして修正したものです。以下で説明しない項目については、このファイルと同じようになっています。

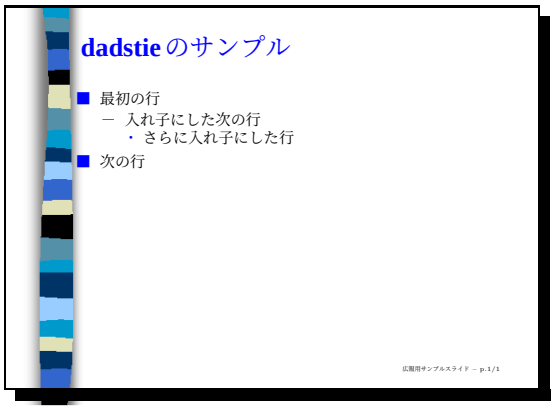

図 12: dadstie スタイル

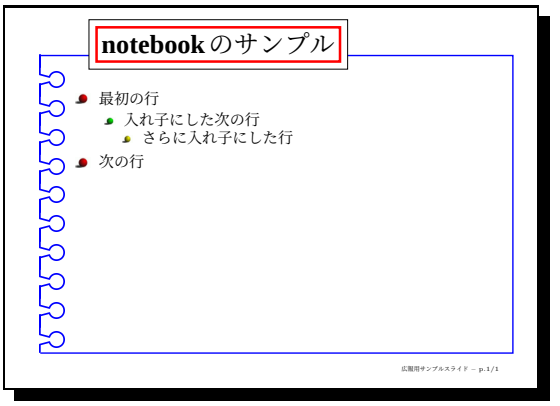

図 13: notebook スタイル

これにより{\anote ab}と入力すると“ ”と表示されます。このフォントは boxnote 環境で使われています。このフォントを使って背景左側を描画します。

タイトルのフォントの色を黒に変更しています。また、タイトルのまわりを\psframebox で2重に囲んでいます。

```
\newcommand{\slidetitle}[1]{%
  \rput[l](0.25,4.5){%
    \psframebox[framesep=5pt, fillstyle=solid, fillcolor=white, linewidth=0pt]{%
      \psframebox[linewidth=2pt, linecolor=red]{\fontTitle{#1}}}}
}
```

外側の\psframebox は白で塗りつぶしています。

フレームは以下のように定義しました。

```
\newcommand{\BasicFrame}[1]{%
  \multirput{90}(-.8, 0)(0, .8){10}{\anote {\blue ab}}
  \psline[linecolor=blue, linewidth=1.1pt](-1.1,-.55)(12,-.55)(12,7.7)(-1.1,7.7)
  \PutLogo % Mandatory
 {#1}}
```

最初の\multirput が左側の部分です。

bluetubes スタイル (図 14)

このスタイルの特徴はタイトル文字です。pst-char パッケージを利用して、文字をふちどりしてみます。タイトルのフォントには、ここまでのスタイルでは和文フォントのゴチック体を使っていませんでしたが、\kanjifamily{gt}を宣言してタイトル部分を gt ファミリ (ゴチック体) にします。kanjifamily は pTEX で利用できるコマンドで、和文フォントのフォントファミリを変更します。このコマンドを使っても欧文フォントに影響はありません。

さらに、タイトルのサイズを大きく (\Huge 相当) して、明るい色にします。

```
\FontTitle{%
  \usefont{T1}{ptm}{b}{n}\fontsize{24.88pt}{18pt}\kanjifamily{gt}%
\selectfont\yellow}{%
  \usefont{T1}{ptm}{b}{n}\fontsize{24.88pt}{18pt}\kanjifamily{gt}%
\selectfont\yellow}
\FontText{%
  \white\usefont{T1}{phv}{m}{n}\fontsize{12.4pt}{12pt}\selectfont}{%
  \white\usefont{T1}{phv}{m}{n}\fontsize{12.4pt}{12pt}\selectfont}
```

このフォントを使ってタイトルを書くわけですが、ふちどりをするために\pscharpath コマンドを利用します。このコマンドを利用するために\RequirePackage{pst-char}を宣言しています。

```
\newcommand{\slidetitle}[1]{%
  \rput[c](5.5,4.8){\fontTitle{%
      \pscharpath[fillstyle=solid, fillcolor=yellow,%
      linecolor=green,linewidth=.6pt]{#1}}}
}
```

塗りつぶしの色 (fillcolor) が字そのもの色で、線の色 (linecolor) がふちどりの色になります。

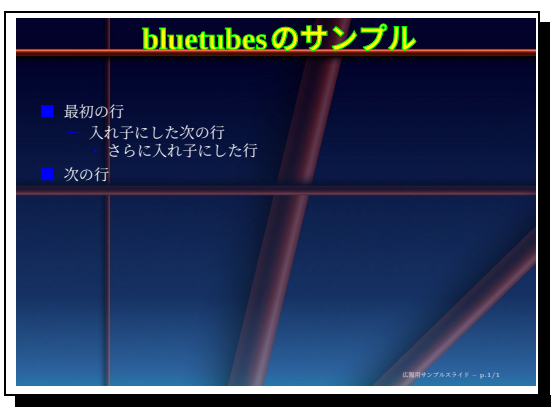

図 14: bluetubes スタイル

図 15: somber スタイル

背景には別に作っておいた画像を使います。

```
\newcommand{\BasicFrame}[1]{%
\rput[lb](-1.8,-1.3){%
\includegraphics[width=14.4cm, height=10.2cm]{bg-blue-tubes}}
 {#1}}

\NewSlideStyle[115mm]{t}{4.3,3}{BasicFrame}
\PDFCroppingBox{10 40 594 800}
```

行頭文字は dadstie と同じです。

somber スタイル (図 15)

このスタイルの特徴は行頭文字です。行頭文字に pifont パッケージの特殊文字を使っています。使える文字の一覧は文献 [3] などを参照してください。

```
\myitem{1}{\red \Pisymbol{pzd}{'064}}
\myitem{2}{\red \Pisymbol{pzd}{'063}}
\myitem{3}{\red \Pisymbol{pzd}{'067}}
```

背景画像とあわせるために、赤い色にしています。

背景画像の読み込み部分などは bluetubes と同じです。

3.4 スタイルの変更

prosper は、背景やタイトルの位置などがスタイルファイルで決められています。したがって、すべてのスライドで同一の見栄えを簡単に作れるわけですが、いくつかのスライドで見栄えをちょっと変えたい場合には少しテクニックが必要です。本稿のの最初のページの図 1 と図 2 は一つの LaTeX ソースですが、本文の位置やフォントが少し異なっています。図 1 では、本文に用いるフォントを小さめにしています。図 2 では、大きなグラフを本文に収めるために、タイトルを少し上に置いています。

このようなことを実現するには、3.2 節で説明したいくつかのコマンドを見映えを変更したいスライドの直前で変更します。例えば、本文の位置をずらすには\NewSlideStyle を、フォントの大きさを 1 枚のスライド全体で変えるには\FontText を呼びだします。

図 1 と図 2 では

```
\FontText{%
  \white\usefont{T1}{phv}{m}{n}\fontsize{12.4pt}{12pt}\selectfont}{%
  \white\usefont{T1}{phv}{m}{n}\fontsize{12.4pt}{12pt}\selectfont}
%
\NewSlideStyle[115mm]{t}{4.3,3}{BasicFrame}
```

と定義してあるスタイルファイルを使っています。図 1 のダイアグラムは大きいので、このスライドを定義した slide 環境の直前で

```
\FontText{%
  \white\usefont{T1}{phv}{m}{n}\fontsize{10pt}{12pt}\selectfont}{%
  \white\usefont{T1}{phv}{m}{n}\fontsize{10pt}{12pt}\selectfont}
%
\NewSlideStyle[115mm]{t}{5.3,4}{BasicFrame}
```

として、本文で使うフォントのサイズを小さくし、本文の位置を右上に変更しています。

このように変更すると、以後のスライドはすべてこの影響を受けるので、このスライドの後で

```
\FontText{%
  \white\usefont{T1}{phv}{m}{n}\fontsize{12.4pt}{12pt}\selectfont}{%
  \white\usefont{T1}{phv}{m}{n}\fontsize{12.4pt}{12pt}\selectfont}
%
\NewSlideStyle[115mm]{t}{4.3,3}{BasicFrame}
```

のように、オリジナルのスタイルファイルの宣言を繰り返します。

> ✔ 現在の設定 (フォントや位置など) を保存しておき、宣言一つで簡単に見映えの切り変えができれば便利ですが、いまのところそのような方法はないようです。

他にスタイルファイルで利用できるコマンドが利用できますが、

```
\newcommand{\slidetitle}[1]{%
  \rput[c](5.25,4.4){\fontTitle{#1}}
}
```

のように\newcommand で指定するものを再定義する場合は\renewcommand にしないといけません。

4 Ghostscript とアウトライン日本語フォント

prosper を使う理由は様々でしょうが、Windows を使いたくないという理由の人は多そうです。しかし、UNIX 上では日本語を含んだスライドで綺麗な表示を可能にする PDF ファイル作成が困難でした。本節では、Ghostscript と日本語アウトラインフォントの利用の仕方について説明します。これを利用することにより、日本語を含んでいる場合でも、UNIX 上でも綺麗な PDF ファイルを生成できます。

UNIX 上で PS ファイルを PDF ファイルへ変換するには、Ghostscript 付録の ps2pdf コマンドを使います。変換する PS ファイルが英語のみで書かれている場合は問題ないのですが、日本語を含む場合、Ghostscript は日本語フォントをビットマップフォントとして埋め込んでしまいます。そのため、解像度に依存した PDF ファイルとなってしまい、きれいな表示が不可能となっていました。従来は、これを解決するには Windows か Machintosh 上で Acrobat Distiller を使って PDF ファイルへ変換するしかありませんでした。

しかし、最近 gs-cjk プロジェクトにより、Ghostscript による PDF ファイルへの変換で、解像度に依存しない日本語の表示を可能にすることができるようになりました。gs-cjk プロジェクトとは、Ghostscript (コマンド名は gs) で日本語、中国語、韓国語フォントを扱えるようにするためのものです。gs-cjk プロジェクトの Web ページ [10] によると

> gs-cjk project は、PostScript 処理系の一つである ghostscript(gs) のフォント処理において、中国語 (繁体字、簡体字)、日本語、韓国語フォントを扱うための機構を開発しています。現在、配布されるパッケージでは、中国語、日本語、韓国語 TrueType フォントを CID-keyed フォントとして使えるようにするとともに、CID-Keyed フォントを使う上での gs の不具合を修正しています。我々は、本家 gs への統合を念頭においた開発を行なっております。

だそうです。

Ghostscript の 6.53 及び 7.05 以降からは、このプロジェクトの成果を含むようになりました。これにより、CID フォントや TrueType フォントを Ghostscript から利用できるようになりました。

Ghostscript は PostScript のインタープリタですので、Ghostscript で利用できるフォントは基本的に PostScript フォントです。しかし、PostScript の和文フォントは高価で、あまり普及しているとは言えません。

上述したように、gs-cjk プロジェクトの努力により、Ghostscript から比較的安価な TrueType フォントや CID フォントが使えるようになりました。これにより、UNIX 上の `ps2pdf` コマンドでも綺麗な和文フォントを含んだ PDF ファイルが生成できるようになりました。もちろん、TrueType フォントや CID フォントを利用できることが前提です。

gs-cjk プロジェクトの Web ページ [10] でも注意していますが、商用の TrueType フォントや CID フォントの利用の際には、ライセンスに注意してください。このページには、商用フォントのライセンスについてのページへのリンクもありますので、一読することをお薦めします。

以下では、古川泰之氏が自身の Web ページ [6] で公開しておられるフリーの東風(こち)明朝と東風ゴチックフォントを利用します。これらは TrueType フォントですが、狩野宏樹氏 [5] により CID フォント化もされています。インストールや設定の説明のために、同じフォントですが、TrueType と CID の両方について説明します。

他にオライリーのサイト[9]で、`WadaMin` などの CID フォントが配布されています。

4.1 Ghostscript のインストール

まず、Ghostscript 7.05 をインストールします。上述したように、GNU Ghostscript の 6.53 及び 7.05 以降からは gs-cjk プロジェクトの成果がとりこまれていますので、特に変わったことはしません。お使いの OS 用のパッケージが用意されている場合は、これを利用しても構いません。フォントのインストール (4.2 節) に進んでください。

> ✔ FreeBSD 上では ports から ghostscript-gnu-7.05_1 をインストールした上で、4.2 節の作業を行いました。これは特に日本語への対応をしていないバージョンです。一方、ja-ghostscript-gnu-jpnfont-7.05 をインストールすれば、東風フォントや CMap ファイルのインストールと東風フォントを使うための設定まで行なってくれます。

ソースファイルからコンパイルする場合にダウンロードするファイルは、以下のとおりです。

- ghostscript-7.05.tar.gz
- gnu-gs-fonts-std-6.0.tar.gz
- gnu-gs-fonts-other-6.0.tar.gz

これらのファイルは、例えば `http://www.ring.gr.jp/pub/GNU/ghostscript/` にあります。以下では、ダウンロードしたファイルはすべて${SOMEDIR} というディレクトリにあるものとします。

インストールには jpeg や zlib などのライブラリが必要ですが、これらはすでにインストール済みであるとします。

ソースファイルを展開後、以下のようにしてインストールします。

[9] `ftp://ftp.oreilly.com/pub/examples/nutshell/cjkv/adobe/samples/`

```
% ./configure
% make
% su
# make install
# cd /usr/local/share/ghostscript/
# tar xzf ${SOMEDIR}/gnu-gs-fonts-std-6.0.tar.gz
# tar xzf ${SOMEDIR}/gnu-gs-fonts-other-6.0.tar.gz
```

デフォルトで/usr/local/share/ghostscript 以下にインストールされます。以降では、このディレクトリを${GS} と表記します。

最後に、4.2 節でインストールするフォントなどの場所を gs_res.ps に書きます。このファイルは${GS}/7.05/lib にあります。このファイルの/GenericResourceDir を、例えば以下のように変更します。

```
/GenericResourceDir (/usr/local/share/ghostscript/Resource/) readonly .forcedef
```

また、このディレクトリを作っておきます。

```
% su
# cd /usr/local/share/ghostscript/
# mkdir Resource
```

4.2 フォントのインストール

次に、TrueType フォントや CID フォントのインストールです。基本的に、ダウンロードしたファイルを展開して、どのファイルを日本語フォントとして使うかを設定ファイルに書くだけです。

gs-cjk プロジェクト [10] から adobe-cmaps-200109.tar.gz, acro4-cmaps-1999.tar.gz の二つのファイルをダウンロードして、CMap ファイルをインストールします。ダウンロードしたファイルは${SOMEDIR} というディレクトリにあるものとします。

ダウンロードしたファイルを/GenericResourceDir で指定したディレクトリで展開します。

```
% su
# cd /usr/local/share/ghostscript/Resource
# tar xzf ${SOMEDIR}/adobe-cmaps-200109.tar.gz
# tar xzf ${SOMEDIR}/acro4-cmaps-1999.tar.gz
```

次に、フォントをダウンロードし、さきほど作った Resource 以下の CIDFont ディレクトリ[10]か、デフォルトのフォントサーチパスにコピーします。デフォルトのフォントサーチパスは gs -h で表示されます。変更していない場合は${GS}/fonts です。

[10]新たに作ります。

TrueType フォントのインストール

TrueType フォントとして、上述したように東風明朝 (kochi-mincho.ttf) と東風ゴチック (kochi-gothic.ttf) を利用します。

```
# cp ${SOMEDIR}/kochi-* /usr/local/share/ghostscript/fonts
```

次に、このフォントを CIDFnmap ファイルに登録します。このファイルは${GS}/7.05/lib にあります。このファイルから同じディレクトリにある CIDFnmap.*というファイル群が読み込まれます。CIDFnmap.*の各ファイルは OS や言語ごとに分かれており、この中で実際のフォントファイルを指定したりします。

CIDFnmap ファイルの “%” で始まる行はコメントで、様々なフォントを利用する場合の実例がコメントとして用意されています[11]。いまから行なう設定は、コメントアウトとコメントインのみで事足ります。

TrueType の東風明朝と東風ゴチックを利用する場合は、CIDFnmap ファイルの

```
%(CIDFnmap.Koc) .runlibfile
%(CIDFnmap.CJK) .runlibfile
```

の行の “%” をはずして有効にします。

CIDFnmap.Koc はデフォルトで CID 化された東風明朝 (ファイル名は Kochi-Mincho) と東風ゴチック (Kochi-Gothic) を利用するようになっています。これらをコメントアウトし、TrueType フォントの東風フォントをコメントインし、以下のようにします。

```
/Kochi-Mincho (kochi-mincho.ttf)   ;
/Kochi-Gothic (kochi-gothic.ttf)   ;
%/Kochi-Mincho  (Kochi-Mincho) ;
%/Kochi-Gothic  (Kochi-Gothic) ;
```

同様に、CIDFnmap.CJK では、

```
%/Ryumin-Light        /Kochi-Mincho          ; % CIDFnmap.Koc
%/GothicBBB-Medium    /Kochi-Gothic          ; % CIDFnmap.Koc
```

のコメントをはずします。これで、TrueType フォントが利用できます。

CID フォントのインストール

CID フォントとして、ここでは東風明朝 (Kochi-Mincho) と東風ゴチック (Kochi-Gothic) を利用します。CID フォントは、先に作った Resource ディレクトリの下に CIDFont というディレクトリを作り、ここにおきます。

[11] 例えば、Solaris や Windows 付属のフォントを利用する例もあります。上述したオライリーのサイトで配布されているフォントは CIDFnmap.Ore に設定例があります。

```
# cp ${SOMEDIR}/Kochi-* /usr/local/share/ghostscript/Resource/CIDFont
```

gs は、デフォルトでこのディレクトリを探すので、設定ファイルなどに書く必要はありません。

CIDFnmap と CIDFnmap.CJK は、上述の TrueType のとおりに変更します。CIDFnmap.Koc は上述の変更をせずに、

```
%/Kochi-Mincho (kochi-mincho.ttf)   ;
%/Kochi-Gothic (kochi-gothic.ttf)   ;
/Kochi-Mincho  (Kochi-Mincho) ;
/Kochi-Gothic  (Kochi-Gothic) ;
```

のようにして利用します。これで、CID フォントが利用できます。

5 その他

この節では、ここまでで取りあげなかったいくつかの話題についてまとめて説明します。

ページサイズ

prosper は A4 サイズを前提に開発されています。一方、Ghostscript ではレターサイズがデフォルトになっています。このため、特にオプションを指定せずに ps2pdf コマンド[12]を利用して PDF ファイルを生成すると、本文やタイトルが意図した位置とずれてしまいます。

以下のようにコマンドラインオプションでサイズを指定するか、

```
% ps2pdf -sPAPERSIZE=a4 file.ps
```

環境変数 GS_OPTIONS を GS_OPTIONS="-sPAPERSIZE=a4"と設定します[13]。

Acrobat Distiller でも同様にレターサイズがデフォルトのようです。[設定]→[ジョブオプション] から [一般] タブを押し、「デフォルトページサイズ」を A4 サイズ (210mm × 297mm) にします。

図・グラフ

単に図やグラフをスライドに埋め込むには、これらを EPS ファイルにしておいて、graphics または graphicx パッケージの\includegraphics コマンドを利用します。

しかし、スライドの背景の色によっては、そのままでは図やグラフの線や文字が見えにくい場合もあります。この場合、color パッケージの\colorbox コマンドなどで背景が白い箱を用意し、この上にグラフを載せる方法が簡単です。図 16 を見ると、グラフ上の線や文字は問題

[12] または ps2pdf13 や ps2pdf14 など。
[13] インストール時のコンパイルオプションで、デフォルトのサイズを A4 に変更しておくことも可能です。

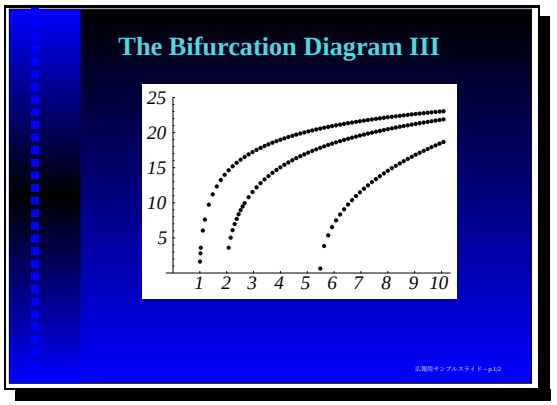

図 16: \colorbox{white} を利用したグラフの読み込み

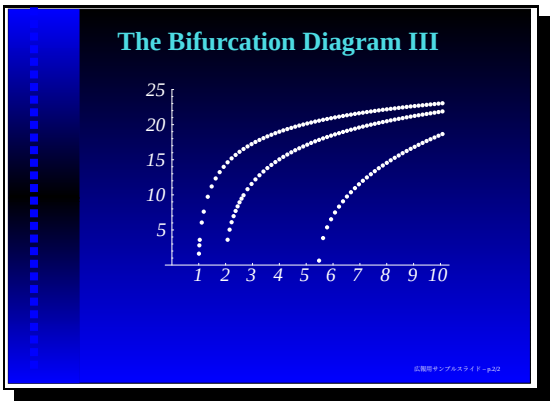

図 17: epsrv を利用したグラフの読み込み

なく見ることができることが分かります。ただし、スライドの背景色と全く違う色になってしまい、見映えがよくありません。

他の方法として http://www.on.cs.keio.ac.jp/~maru/epsrv/ で紹介されている epsrv パッケージを使う方法があります。このパッケージを利用すると、EPS ファイルの黒い線やテキストを白くしてくれます (図 17 参照)。グラフの背景は透明になっていて、スライドの背景がそのまま見えるのが分かります。このスライドは、プリアンブルで\usepackage{epsrv} した上で、

```
\epsrv{\includegraphics[width=8.5cm]{file.eps}}
```

と入力しています。ただし、任意の色の変換ができるわけではなく、黒を白に変換するだけです。

ノードと重ね合せ

pst-node パッケージを利用すれば、LaTeX の要素に名前をつけてノードとし、ノード間に接続線 (直線や矢印など) を引くことができます。例えば、\rnode{a}{ノード a} とすれば、「ノード a」を “a” という名前のついたノードにすることができます。他に、ノード部分を円で表わす\cnode(x,y){半径}{名前}や、文字列を円で囲んだ\circlenode{名前}{テキスト}などがあります。

複数のノードに名前をつけたら、\ncline{a}{b}としてノード間に線を引くことができます。\psline と同様にオプション引数 [*arrows*] を指定すれば、ノード間の線も矢印にすることができます。他に、様々な種類の線でノード間を結ぶことも可能です。詳しくはマニュアル [13] を参照してください。

この機能と\overlays コマンドを組み合わせてみます。やりたいことは、最初のページ[14]にノードとなるテキストを置き (図 18)、次のページで別のノードとこれらのノード間に線を引

[14] ここでは 1 枚のスライドが\overlays で囲まれている場合は、このスライドは複数の “ページ” から成るとします。

く (図 19) というものです。つまり、パワーポイントでいうアニメーション機能に相当するこ

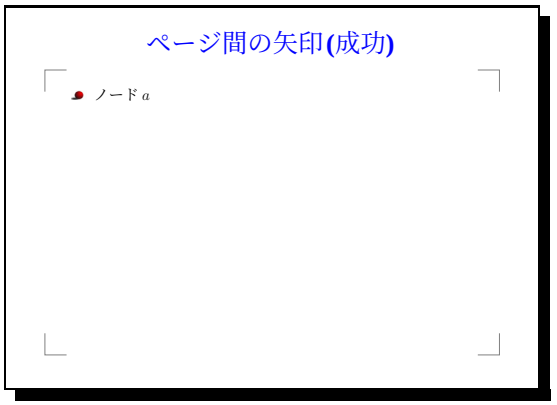

図 18: ノードを 1 つ描画

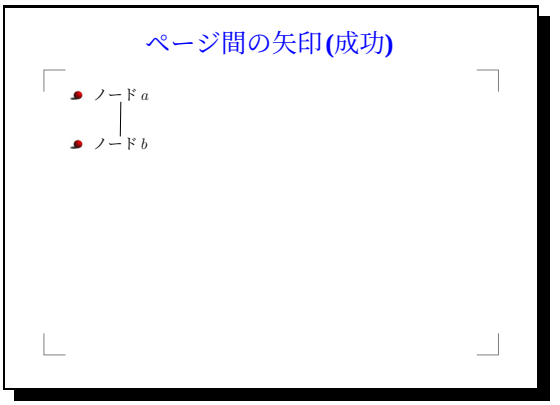

図 19: 別のノードを描画し、さきほどのノードとの間に線を引く

とと、ノードの描画を組み合せて使いたいわけです。

これを実現するために、単純に以下のようにしてみます。

```
\overlays{2}{
  \begin{slide}{ページ間の矢印 (失敗)}
    \begin{itemize}
    \item \rnode{a}{ノード$a$}
      \fromSlide{2}{
    \item[] % 行間を空ける
    \item \rnode{b}{ノード$b$}
      \ncline{a}{b}} % ノード間の直線
    \end{itemize}
  \end{slide}}
```

次のノードと直線の描画部分を\fromSlide に中に入れています。しかし、実はこれでは不十分で、このように入力した場合は図 20 と図 21 のような結果になります。図 20 において、既に直線が引かれています。この時点では、片方のノードがないので無関係な方向に線が引かれたようです。

図 18 と図 19 のようにするには、

```
\overlays{2}{
  \begin{slide}{ページ間の矢印 (成功)}
    \begin{itemize}
    \item \rnode{a}{ノード$a$}
      \fromSlide*{2}{ % <--- ここがポイント
      \item[]
      \item \rnode{b}{ノード$b$}
      \ncline{a}{b}} % ノード間の直線
    \end{itemize}
  \end{slide}}
```

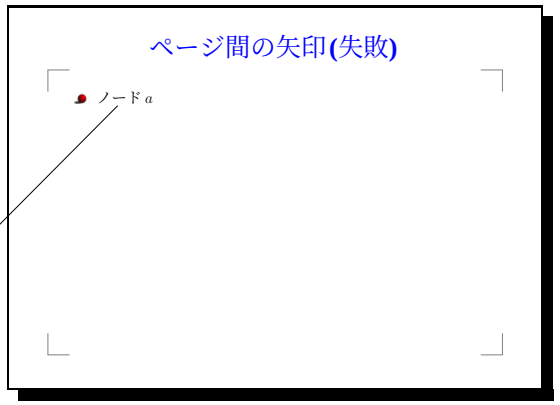

図 20: ここで不正な線が引かれてしまう

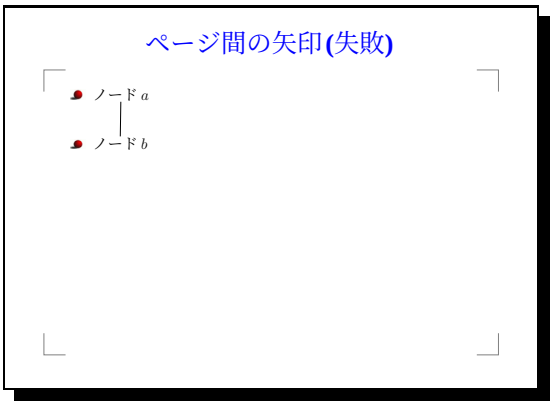

図 21: 別のノードを描画し、さきほどのノードとの間に線を引く

とします。つまり、\fromSlide ではなく\fromSlide*を使います。

表とノード

prosper では表も問題なく利用できます。さらに pst-node パッケージを利用すれば表の各要素を矢印で結ぶことも可能です。実際、図 1 は tabular 環境を使って描画しています。表の各要素に\rnode コマンドで名前をつけ、

```
    \ncLine{a}{c}$\lput{:L}{\textrm{計算}}$
```

のようにして矢印を書いています。

以下に、図 1 の全ソースをつけます。

```
\begin{tabular}{cc}
  \multicolumn{1}{c}{\textbf{連続系}}  & \textbf{離散系}\\[.4cm]
  \rnode{a}{$G(u,u_x)$} & \rnode{b}{$G_d(U^{(n)})$} \\[1.5cm]
  \rnode{c}{変分導関数} & \rnode{d}{離散変分導関数} \\[1.5cm]
  \rnode{e}{偏微分方程式} & \rnode{f}{差分方程式} \\[1cm]
  \psset{nodesep=3pt,arrows=->, linewidth=.4pt, linecolor=white}
  \ncLine{a}{c}$\lput{:L}{\textrm{計算}}$
  \ncLine{a}{b}$\lput{:U}{\textrm{\red 近似}}$
  \ncLine{b}{d}$\lput{:L}{\textrm{\red 計算}}$
  \ncLine{c}{e}$\lput{:L}{\textrm{代入}}$
  \ncLine{d}{f}$\lput{:L}{\textrm{\red 代入}}$
\end{tabular}
```

各要素は\rnode や\titledframe で囲まれた複雑な数式なので、ここでは各要素を簡略化して書いています。

表の各要素はボックスで囲んでいますが、以下のように定義した\titledframe コマンドで描画しています。

```
\usepackage{fancybox}
\newcommand{\titledframe}[2]{%
  \boxput*(0,1.5){\psframebox[linestyle=none,fillstyle=none]{\yellow #1}}%
  {\psframebox[linecolor=yellow, framesep=4pt]{%
  \parbox{.37\slideWidth}{\centering#2}}}}
```

ここでは、テキストを枠で囲むために fancybox パッケージを利用しています。

\fromSlide などのページの重ね合せと一緒に表を使うこともできます。これを使えば、各要素を順にアニメーション表示していくこともできます (図 22 と図 23 参照)。ただし、列の区

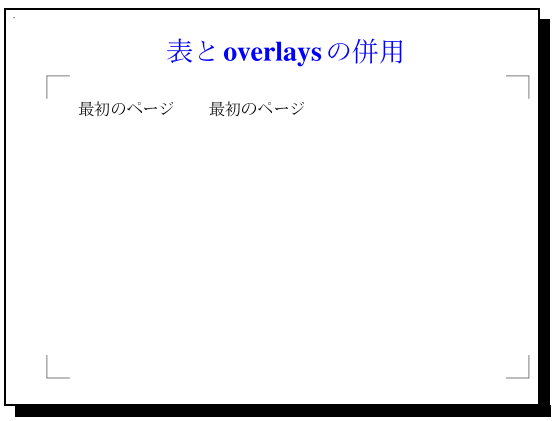

図 22: 表の最初の行の要素全部を表示

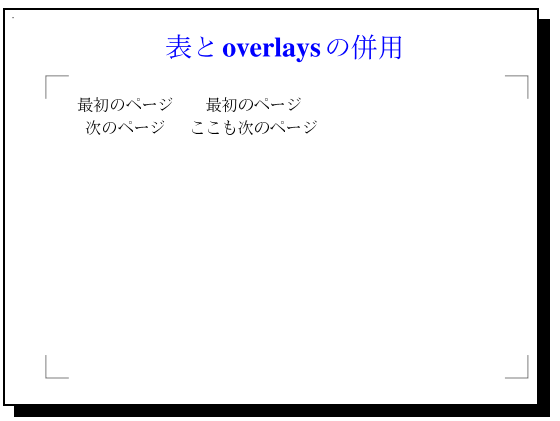

図 23: 次の行の 2 要素を同時に追加表示

切りを示す “&” をまたいで使うことはできないため、1 行をまとめて\fromSlide の引数にすることはできません。その代りに、

```
\fromSlide{2}{a} & \fromSlide{2}{b} \\
```

などとします。図 22 と図 23 は以下のようにして作りました。

```
\overlays{2}{
  \begin{slide}{表と overlays の併用}
    \begin{tabular}{cc}
      最初のページ                & 最初のページ \\
      \fromSlide{2}{次のページ}& \fromSlide{2}{ここも次のページ}
    \end{tabular}
  \end{slide}
}
```

2 行目の各列の要素を独立に\fromSlide の引数にしています。もちろん、各列の要素をすべて同じページに表示させる必要はなく、

```
\fromSlide{2}{a} & \fromSlide{3}{b} \\
```

などとして、列を順々に表示させることも可能です。

tabular 環境だけでなく、array 環境や eqnarray 環境でも同様に問題なく使えます。

最新開発版

prosper の最新開発版は CVS で取得可能です。http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/prosper/prosper/ からダウンロードしてください。メインのクラスファイルである prosper.cls はバージョン[15]が 1.5 になっています[16]。様々なバグフィックスがなされているそうでが、特に\label と\ref による相互参照ができるようになったのが有難いです。

最新版を利用するには、上記 URL からファイルをダウンロードし、既存の prosper.cls といれかえるだけで使えます。

6 おわりに

2 回にわたり prosper の解説をしました。prosper が提供している機能の説明としては、これだけで十分だと思います。しかし、実際にプレゼンテーション用の資料を作ると、様々なフォントを使いたかったり、数式をより複雑に配置したかったりします。このような場合には prosper に関する知識というより、広く LaTeX や TeX、関連のパッケージやコマンドの知識が必要になります。

例えば LaTeX から図を書く方法として、本稿では PSTricks を説明しましたが、PSTricks の機能だけでも他にたくさんあります。文献 [9] には、PSTricks も含め他の描画系パッケージについても詳しく説明してあります。

プレゼンテーション用のスライドでは、フォントも重要な要素です。フォントの使い方については上述の文献 [9] や [3] を参照してください。特に文献 [3] は著者が日本人で、和文フォントについての記述もきちんとあります。

また、文献 [3] の著者である奥村先生は TeX に関する Web ページを開設しておられます [4]。ここにはフォントやパッケージ、LaTeX のエディタなど広く情報が集まっています。質問用の掲示板もあり非常に参考になるページです。

筆者も prosper に関する Web ページを開設しています [1, 7]。本稿で作ったスタイルファイルは [1] に置いています。[7] では、Windows 環境での prosper の使い方を中心に説明しています。参考にしてください。

参考文献

[1] 池田 大輔、 LaTeX class file: prosper, http://qatm.cc.kyushu-u.ac.jp/~daisuke/prosper/index.html

[2] 池田 大輔、渡部 善隆、 prosper を使おう—LaTeX でプレゼン資料を作成—、 九州大学情報基盤センター広報、Vol. 2, No. 1, pp. 41–62, 2002.

[15] ファイルの先頭付近にある\Prosper@Version のすぐ後の文字列です。

[16] 安定版というべき prosper-1.00.4.tar.gz に付属しているのはバージョン 1.1 です。

[3] 奥村 晴彦、 [改訂版]LaTeX–美文書作成入門—、 技術評論社、2000.

[4] 奥村 晴彦、 日本語 TeX 情報、 http://www.matsusaka-u.ac.jp/~okumura/texfaq/

[5] 狩野 宏樹、 東風明朝 CID 化キット、 http://kappa.allnet.ne.jp/Kochi-CID/

[6] 古川 泰之、 My Linux 日本語化計画、 http://www.on.cs.keio.ac.jp/~yasu/jp_fonts.html

[7] 渡部 善隆、 Prosper の使い方 – A LaTeX class for writing transparencies –, http://www.cc.kyushu-u.ac.jp/RD/watanabe/RESERCH/PROSPER/index.html

[8] T. Esser, The teTeX Homepage, http://www.tug.org/teTeX/

[9] M. Goosens, S. Rahtz, F. Mitteibach 著: 鷺谷 好輝 訳、 LaTeX グラフィックスコンパニオン、 アスキー、2000 年。

[10] gs-cjk プロジェクト http://www.gyve.org/gs-cjk/index-j.html

[11] MagicPoint, http://www.mew.org/MagicPoint/

[12] The MagicPoint Gallery, http://puchol.com/cpg/software/mgp/

[13] T. Van Zandt, PSTricks: PostScript macros for Generic Tex. User's Guide, http://www.tug.org/applications/PSTricks/.