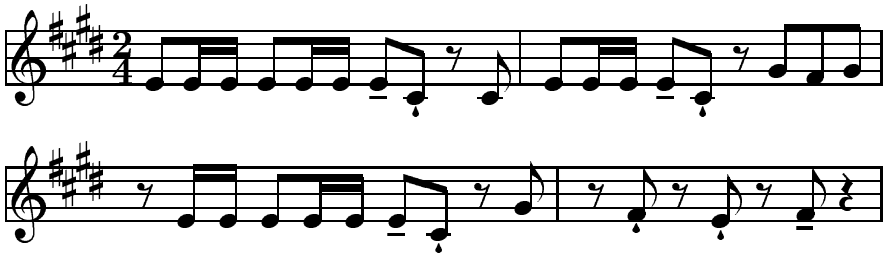
Theoretical Science Group

理論科学グループ

部報 298.5 号
— 夏合宿号 —

目　次

いくぶん実用的な記事 **1**
夏合宿の予定表 . . . . . 【Ktya】 1

そのほか一般記事 **3**
〜Herbert の Solver について〜 . . . . . 【Ktya】 3
MIDI と楽譜 . . . . . 【molcul】 9
『すごい眠いんだが、明日バイト遅刻しないといいな』 . . . . . 【過密都市】 12

いくぶん実用的な記事

夏合宿の予定表

Ktya

夏合宿の予定について大まかにここで説明しておきます.

一日目

8:30	集合 新宿西口改札
9:00–30	出発 JR 湘南新宿ライン
9:30–10:00	大宮着 (途中下車
	(大宮公園, 交通博物館等をまわります)

大宮公園について

http://www.go2park.net/parks/oomiya.htm

交通博物館について

http://www.railway-museum.jp/top.html

(荷物が大きい人はこの辺はすいません)

13:16	大宮駅発 (JR 東北本線)
	ご飯はテキトーに済ませる？
14:34	宇都宮
15:27	黒磯
16:43	郡山 (JR 磐越西線・会津若松行)
17:26	翁島
17:40	宿着, 飯

宿周辺の地図・宿の HP

http://tamanoyu.net/

以降自由時間

二日目 (ややガチ)

10:00　宿発
10:31　翁島発
11:01　会津若松
　　・ぶらぶら
　　・お昼は会津若松駅前点の来夢ラーメン
13:11　会津若松発 (JR 磐越西線・郡山行)
13:40　猪苗代駅着
　　・遊覧船とか
14:55　五色沼へ (バス)
15:18　五色沼入り口
　　五色沼まわる
宿へ, ご飯後は自由時間です

三日目 (どっかぶらぶら)

ちゃんとは決めてませんが、どっかをぶらぶらする予定です ()

そのほか一般記事

～Herbert の Solver について～

Ktya

なにを書こうか非常に迷ったのですが夏合宿号と言う事でテキトーにコードを晒す事にします. 内容については半年程前に書いた Herbert Online Judge の (部分的な) ソルバーです. 記事にする程の工夫は全くしていないのですが, 記念と言う事で許していただけると嬉しいです w というか今も制作途中でもうちょっとバリエーション増やせたらいいなぁと思っています.(それらを今後の記事に載せるかは不明) 検索したものは

$$a(X) : Ya(Z)$$
$$a(W)$$

という形のものです. 方針としては

- 全探索
- rl,lr,rrr,lll といった無駄なコードを省く
- ある程度動く範囲を指定する (灰色マスに入ったら打ち切る、など. 以下で乗せるコードはそれを実行してません.)
- 数字からコード生成
- その他, 動く step 数の設定等.

方針として悩んでいる所もまだまだ多いのでメモ程度です, もっと高速に動くのを一応目指してます.

以下コード.C++です. アドバイスとか頂けたら嬉しいです.

```
#include<iostream>
#include<string>
#define M 50000

using namespace std;
int dx[4]={0,1,0,-1};
int dy[4]={-1,0,1,0};
int startx,starty;
bool skip;
```

～Herbert の Solver について～

```
bool hantei[25][25];
string hoge[15]={"","s","r","l","ss","sr","rs","sl","ls","rr","sss",
"ssr","ssl","srs","sls"};
string field[25];
string hosoku[3]={"","r","rr"};
// judgesystem
// code tenkai
// search
//x ↓ y →  string[x][y]
void partition(string str){//処理しやすく分割

    for(int i=0;i<25;i++){
        field[i]="";
        for(int j=0;j<25;j++){
            if(str[i]=='u')startx=i,starty=j;
            field[i]+=str[i*25+j];
        }
    }

}
void init(){//灰マスを踏んだときの初期化
    for(int i=0;i<25;i++){
        for(int j=0;j<25;j++){
            hantei[i][j]=(field[i][j]=='o'? false:true);
        }
    }
}

bool judge(string code){//ジャッジシステム
    int x=startx,y=starty,dir=3;
    bool fflg=false;
    init();
    for(int i=0;i<M;i++){
        switch (code[i]){
          case 's':
            if(x+dx[dir]<25&&x+dx[dir]>=0&&y+dy[dir]<25&&y+dy[dir]
>=0){
```

```
                if(field[x+dx[dir]][y+dy[dir]]!='x'){
                    x+=dx[dir];
                    y+=dy[dir];
                    if(field[x][y]=='*')init();
                    if(field[x][y]=='o')hantei[x][y]=true;
                }

            }
            break;
          case 'r':
            dir=(dir+3)%4;
            break;
          case 'l':
            dir=(dir+1)%4;
            break;
        }
        bool flag=true;
        for(int i=0;i<25;i++){
            for(int j=0;j<25;j++){
                if(!hantei[i][j])flag=false;
            }
        }
        if(flag)fflg=true;
    }
    return fflg;
}

string henkan(int a){//整数から文字へと変換（全探索のため）
    string s="";
    int b;

    while(a>0){
        b=a%4;
        switch(b){
          case 0:
            s=s+"r";
            break;
```

```
            case 1:
              s=s+"l";
              break;
            case 2:
              s=s+"s";
              break;
            case 3:
              s=s+"X";
              break;
          }
          a=a/4;
      }
      //以下例外をはじくための skip を設定
      if(s.find("X")==string::npos)skip=false;//X が無い文字列を生成して
チェックしても意味が無いので飛ばす
      if(s.find("rl")!=string::npos)skip=false;//rl は意味が無い
      if(s.find("lr")!=string::npos)skip=false;//lr は意味が無い
      if(s.find("rrr")!=string::npos)skip=false;//rrr は意味が無い
      if(s.find("lll")!=string::npos)skip=false;//lll は意味が無い
      if(s.find("ll")!=string::npos)skip=false;
      return s;
}

string henkan2(int a){//上に同じ
      string s="";
      int b;
      while(a>=3){
          b=a%3;
          if(b==0){
              s=s+"s";
          }else if(b==1){
              s=s+"r";
          }else{
              s=s+"l";
          }
          if((a-b)==0)break;
          a=a/3;
```

```
    }

    if(s.find("rl")!=string::npos)skip=false;//rl は意味が無い
    if(s.find("lr")!=string::npos)skip=false;//lr は意味が無い
    if(s.find("rrr")!=string::npos)skip=false;//rrr は意味が無い
    if(s.find("lll")!=string::npos)skip=false;
    if(s.find("ll")!=string::npos)skip=false;
    return s;
}

string tenkai(string str1,string str2,string str3){//与えられたコードから
実際に実行するものを展開する
    string ret;
    string temp1,temp2,x=str3;
    int cnt=0;
    while(ret.size()<M){
        temp1="";
        temp2="";
        for(int i=0;i<str1.size();i++){
            if(str1[i]!='X')temp1+=str1[i];
            else temp1+=x;
        }
        ret+=temp1;
        for(int i=0;i<str2.size();i++){
            if(str2[i]!='X')temp2+=str2[i];
            else temp2+=x;
        }
        x=temp2;
        cnt++;
        if(cnt>M)break;
    }
    return ret;
}


int main(){
    string str;
```

```
    cout<<"フィールドの文字と検索したいステップ数を入力してください:";
    cin>>str>>M;
    partition(str);

    string one,two,three,x,y,z;


    for(int i=0;i<4096;i++){
        for(int j=0;j<4096;j++){
            for(int k=0;k<15;k++){
                skip=true;
                one=henkan(i);
                two=henkan(j);
                three=hoge[k];

                for(int t=0;t<3;t++){
                    for(int s=0;s<3;s++){
                        x=one+hosoku[t];
                        y=two+hosoku[s];
                            if(x.find("lr")!=string::npos)skip=false;
                            if(y.find("lr")!=string::npos)skip=false;
                        if(skip){
                            if(x.size()+y.size()+three.size()==7&&thr
ee.size()<3){
                                cout<<x<<" "<<y<<" "<<three<<endl;
                                if(judge(tenkai(one,two,three))){
                                    cout<<"clear"<<endl<<"a(X):"<<x<<
"a("<<y<<")"<<","<<"a("<<three<<")"<<endl;
                                    return 0;
                                }
                            }
                        }
                    }
                }
            }
        }
    }
```

```
    cout<<"fail"<<endl;
    return 0;


}
```

MIDI と楽譜

molcul

Attention!

- 編集長の尺埋め記事第 2 弾です.
- これは前回から続いてますが、読んでいなくても大丈夫です.
- 音楽に興味がない人は読めない (=読む気を失くす) と思います.

手続き論であって, 実際のプログラミング言語で実装したり, 具体的に MIDI ファイルを書いたりしていません. 前回：

> 音符と曲想記号の変換について書いていく予定です. その際, MusiXTEX という楽譜作成用の TEX を使います.

とか書きましたが, いちいち楽譜を打つのは面倒なので, 編集権限で表紙に楽譜を 1 つ載せるに留めておきます.

音符の発音

主な作業は, 譜表を段ごとに左からずっと読んでいって, 音符の種類と位置から音高と音長を読み取り, MIDI メッセージに出力する, ということになります. 音符には強度 (velocity, Vel) といわれる, 個別の音の強さが設定できますが, 大抵 100 とされることが多いです.

発音メッセージ | 発音時間 | 止音メッセージ | 休止時間 |

で 1 まとまりです. [1] いわゆる音符の長さは (発音時間)+(休止時間) で与えられ, 音価 (note value, NV) と言われます. 発音時間はゲートタイム (gate time, GT) と言われ, 通常の楽器では音価の 90%か 95%くらいにされます.

例えば, 四分音符の長さは 480 とされるため, GT が 90%NV なら発音時間は 432, 休止時間は 48 となります.

[1] 本当は時間の後にメッセージが来るのだが, 上図のほうが見やすい.

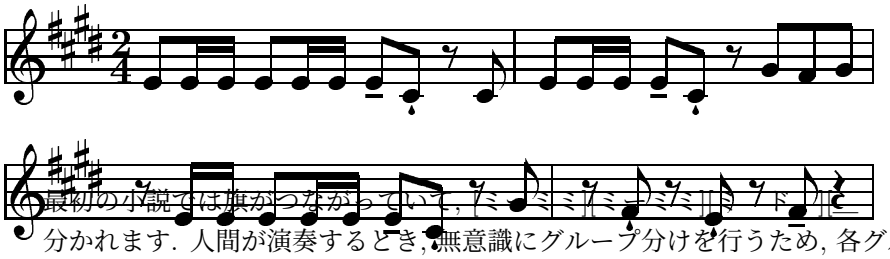
音符の装飾

音符にはいろいろな装飾記号が付きます. 表紙にも点や線の付いた音符があるでしょう. これを実装するのは割と簡単です.

スタッカート	GT を 50%NV にする.
テヌート・レガート・スラー	GT を 105%NV にする. 同じ音が連続する場合
アクセント	Vel を 120 程度にする
トリル	音程 (Pitch Bend, PB) コントローラ [2] を連続的に変化させる.

グルーピングとヒューマナイズ

実際, 人間は楽譜を書いてある通りには演奏しません. 例えば, 表紙の楽譜を見ましょう：



ド] と, 4 つの部分に分かれます. 人間が演奏するとき, 無意識にグループ分けを行うため, 各グループが時間的にまとまりやすくなります. 大げさに言えば「ミーミミ □ ミーミミ □ ミ ド □＿ ド 」というように, グループとグループの間にわずかな間隔ができます. さらに, 「**ミ**ーミミ**ミ**ーミミ」と, グループの頭が若干強くなります.

さらに, 人間は機械ではないので各パラメータはばらつきます. コンピュータでもそれを乱数をもちいて生成することがあります. これをヒューマナイズといいます. いずれも, かならず処理しなければいけないわけではないオプションです. 人間の演奏している音ではない, 機械の生成音を表現したい場合はこれらの処理は不要です.

調・拍子

この 2 つは MIDI ファイルで直接記述します. これは楽譜データから取り込んで直接記述するだけです. 例えば「4/4 拍子変ロ長調」は”00 ff 58 04 04 02 18 08 00 ff 59 02 fe 00”となります.

テンポ・強弱

テンポは MIDI ファイルに直接記述, 強弱は MIDI の表情 Expression コントローラの調整という違いがありますが, この二つはよく似ています. 音楽標語はあくまで目安なので, 変換するとき

は独自基準を設けるといいでしょう. 以下は自分の基準です：

テンポ — Andante(遅め, BPM 72)・Moderate(中間, BPM 108)・Allegro(速い, BPM 144)
強弱 — f(強い, 112)・p(弱い, 80)

rit.(だんだん遅く) や *cresc.*(次第に強く) などは曲によって理想とする変化率が違いますが, 指定された範囲内で 20%程変化させるのが一般的でしょう.

また、小ネタですが, フェルマータ (音を適切に伸ばす) は, 0.5 秒ほどテンポを半分にするとうまく実装できます.

その他のコントローラ

その他にも, MIDI には様々なコントローラが存在します. 多くは楽譜に明示されていないパラメータで, 使用される楽器などを仮想して適切に設定してやる必要があります.

音量 Volume	強弱ではなく楽器自体の音量で、通常演奏中は一定にしておきます.
音像 PanPot	ステレオの左右位置で, 楽器ごとにずらしておくと自然な音楽になります.
反響 Reverb	音の残響を変えます. 実際はそれほど変化しません.
合唱度 Chorus	音の広がりが変わります. 大きいほどその楽器が数が多いことを示します.
揺れ深さ Modulation	音の揺れの深さで, 大きいほど不安定な音になります.
etc...	

最後に

他にも考慮しないといけないパラメータはいっぱいありますが, 自分としては楽譜を MIDI に投射するときの主なポイントをまとめました. 紙の楽譜から目で写すのもいいですし, 殊勝に楽譜のベクターデータから MIDI を作成するのもいいかもしれません. (自分ではしません)

次回, 逆に MIDI から楽譜を作ることについて何か書くかもしれません.　書かないかもしれません.

『すごい眠いんだが、明日バイト遅刻しないといいな』

過密都市

★ はじめに

高校の友人と定期的に Skype チャットをするのですがある時マンガについての話になりました。彼は『咲-Saki-』が好きで阿知賀編が今年春にアニメになったのをきっかけに私に勧めてきたのでした。実は私は『ムダヅモ無き改革』とか読んだこと無くて麻雀マンガのことは知らないのですが本編アニメを見た感想として、麻雀の運ゲの部分を逆手にとって壮絶な超展開を繰り広げたり能力者バトルに仕立てたりで対局を面白く表現し、また麻雀以外の部分の展開もきちんと設けていてよく出来たマンガだと思いました。さて彼とはその流れでその他もろもろのマンガの話になったわけですが、

私「競技プログラミングもののマンガって本当何で無いんだろうなと思うよ」
彼「え？そんなんあるわけないでしょｗ」
私「……」

プロコンのマンガがあったっていいだろ！いい加減にしろ !!!! 何故です？確かに野球とかサッカーとかよりもマンガになりにくそうだというのは分かりますけどサイクリングはおろか競技クイズに百人一首、挙句の果てにはオーバークロックなんてニッチなジャンルまでマンガになってんですよ !? なんでプロコンがマンガになってないんだ。競技プログラミングがマンガにならないその幻想をぶち壊す！

★ 障害

競技プログラミングが未だマンガにならない理由には以下のものがあると考えます。

1. 「競技プログラミングがニッチなジャンルであること」
2. 「競技プログラミングはビジュアル的には地味であること」
3. 「競技プログラミングにはそれなりの知識が伴うこと」

ただし 1 の障害は大したものではないと私は考えます。実際上に挙げたようなニッチなジャンルだって立派にマンガになっている例もあるわけですし、マンガの重要なファクターはテーマがメジャーであることではありません。
「ドラマは見るけどアニメは子供の見るものでしょ」とか逆に「アニメは見るけどドラマとか見んわｗｗｗ」という人が世にはいる気がしますが、私はドラマもアニメも何らかのストーリーを展開する娯楽という点で同じものだと思っています。小説やマンガも表現の仕方が異なるだけで同じと思っています。この類の娯楽の良し悪しは消費者に影響を与えるものを持つかどうか、ストーリーから伝わってくるものがあるかどうかで決まると私は思っています。（ただギャグものに

図 4.1: ↑『87CLOCKERS』(二ノ宮智子)　液体窒素など様々な手で CPU の計算処理能力を限界まで高める「オーバークロック」に挑戦する音大生を描いたぶっとんだマンガである

関して言えばそれが本当に面白いかどうかという純粋な娯楽の役割が重要になるでしょうが。）そしてこれは当然扱うジャンルの知名度によりません。私は単純なことを言ってしまえば、メジャーかマイナーかに関わらずあるジャンルに対して人間が真剣に取り組んでいる姿を描ければ（それは難しいことではあるだろうが）簡単に影響力を持ちうると思っています。人間の真剣な姿から何故だか我々は元気や感動を貰うことができるからです。逆にニッチであるからこそそういう姿が描ければ今まで知られていたことが少なかった分、多くのことを伝えることができるのではないのでしょうか。

ということで以下では 2、3 の障害をいかにして扱うか考えていきたいと思います。ジャンルがニッチ過ぎて描き手がいないなんてのは知りません。同人ってものがあるんだから何とかしてください。

★ 構成

競技プログラミングのマンガを描くとしたらギャグマンガという線もありますが、展開重視のストーリーマンガの方が競技プログラミングの競技である側面を生かせると思います。また、上で述べたような競プロに精を尽くす人間像を描くことを目的として登場人物はやはり高校生あたりがよいと思われます。登場人物はやはり若い方が話が作りやすいし、主人公が高校生ならば上（大学生など）にも下（中学生など）にもライバルを作りやすいです。競争相手は出来るだけ多様な人々が揃っていたほうが面白いでしょうから。登場する人物の性別のパターンとして考えられるのは「ほとんど男」（テニプリ、イナイレ etc)、「男女混合」（ちはやふる、ナナマルサンバツ

etc)、「ほとんど女」（ハヤテ、咲 etc）の３つがありますが、競プロで「ほとんど男」というのはまずウケないというのは同意してもらえるでしょう。競技プログラミング勢的には（完全に個人的なイメージですが）「ほとんど女」パターンが求められているのかなとも思いますが、私は「男女混合」を推します。まあどちらでもいいでしょう。ストーリーの軸ですが、これはマンガの特徴になる部分なので今特にここで扱う必要はないと判断します。ただ物語の基本として、登場人物がものごとに巻き込まれ変遷し最後にそこから出る、その一連の流れがストーリーであるので最初主人公は競プロをやっていないもしくはそれに近い状態の方が面白くなると思います。よくある「あなた○○部に入らない？」展開ですね。また、単に競技プログラミングを極めることではない別の目的が主人公にあると面白いでしょう。『咲-Saki-』なんかだと「数年前に家庭の事情で離れ離れになった姉と仲直りをする」とかですかね。そこを考えるのが非常に難しいのでしょうけど。

図 4.2: ↑『ナナマルサンバツ』(杉基イクラ)　競技クイズを題材にしたマンガ

さてここで２つの障害について考えたいと思います。マンガとは言わずもがな専ら絵によって状況を描写し話を展開していくものです。ですからここに障害２「競技プログラミングはビジュアル的には地味であること」が立ちはだかってきます。現実の競技プログラミングでは競技者はサッカー等のように激しく動くことはなく、ただいすに座って頭をひねりキーボードを打つことしかしません。なんかの奥義の必殺技が決まっている様子も描けません。そのいってる様子を単に絵で表現するだけでは当然地味なものになります。そのため競技中の見せ方に工夫が必要となります。

図 4.3: ↑ドラゴンボール（鳥山明）より　かめはめ波が決まっている所

★ 対策

対策としてはまず競技者だけではなく競技の観戦者の視点を作ることで一つのコンテストを出来るだけ立体的に見せることが考えられます。動いていないものでも視点を変えていけば生き生きと表現できるはずということです。やはり競プロで動くのは体ではなく頭なのですから各人の思惑などを交えていくのが普通なのではないでしょうか。観戦者を入れることで問題の解説などもしやすくなりますし。またこれと同じ観点から、リアルタイムで更新されるスコアボード的なものや TopCoder でいうような Challenge などの存在は重要視するべきだと思います。単に問題を解いて答えあわせしてもらい点数を競うというのはそれはそれで確かに競っていることになるわけですが、それでは競技者それぞれが隔絶されていることになり各人の思惑の交差もへったくれも無くなるわけです。スコアボードや Challenge は競技者同士が干渉しあう場を提供するわけでここに駆け引きを作ることが可能になるわけです。あとプログラミングの形式をペアプロにするというのも考えられます。一人で問題に取り組んでいる様子よりもチーム、またはペアで取り組んでいるほうが 1 つのコンテストを表現する視点は多様化します。またペアプロになるといかに相方と協力して効率よく問題を解いていくかという戦略がプログラミング自体に生まれてくるので話が面白くなるのではないでしょうか。視点変更などではなく単純にビジュアル面を補強したいというならば、マンガならではの擬人化などを使っていけばいいと思います。『BLOODY MONDAY』という高校生ハッカーがテロ集団と対峙するマンガがありますが、その中で主人公のハッキングを描くときにはハヤブサがビジュアルとして使われることがあります。様々なトラップや鍵が仕掛けられた電脳世界を主人公が操るハヤブサが飛び抜けていくビジュアルを比喩的に用いてセキュリティを破っていく様子を描写しているわけです。競プロのマンガでもキャラが書いたプログラムがそれぞれのキャラに対応した姿となって問題ケースを撃破していく描写が偶に描かれるだけ

でも良いアクセントになるのではないでしょうか。

最後に障害 3「競技プログラミングにはそれなりの知識が伴うこと」について考えていきたいと思います。そもそもプログラミング自体文系の方にはあまりなじみがないようで、少し注意をしないと全く何やってるのかわからないマンガとなってしまいます。そもそもマンガの対象層をプログラミング経験者に絞るというのも手だとは思いますがそれではもったいないし、出来るだけ多くの人に競プロマンガを読んでもらいたいものです。

しかしこの分かりにくさを解消するのはなかなか至難の業だと思われます。究極の対策法としては「競技プログラミングをやっている様子は描かれるがその中身にまでは触れず、人間関係など別のファクターで話を展開する」という方法が考えられると思います。『けいおん!』などはこの方法で描かれていて、彼女らは確かにバンドはしているけれども主に描かれているのは日常パートです。この方法で確かに多くの人々にとって分かりやすいマンガを描くことはできるでしょうけれども、それが競プロマンガかといわれると微妙な気がします。大衆に受け入れてもらうために逆にプログラミング経験勢を引き込む魅力を失ってしまっている気がします。しかし毎回毎回アルゴリズムを解説するようなコーナーなり展開なりが出てきてしまっていてもそれはくどいし、そもそも大衆に理解してもらえるかどうかわからないし、マンガでない別のところでやればよいことです。かといって全くアルゴリズムについての解説を入れないとやはり分かりにくいものになってしまうため、オーダーについてやＤＰ、枝狩りなどの基本的な部分に解説をとどめるなど工夫が必要となるでしょう。またとっつきにくさを感じさせないように Warshall–Floyd などの具体的な名前はむやみに出さないほうが良いでしょう。私に今考えられる最も有効な対策としては、マンガ内で出題される問題は、決まった答えがあるものではなくできるだけ効率化を図るタイプの問題を多くするというのが挙げられます。効率化を図るタイプの問題では誰でも分かる単純な工夫が鍵になることもおおいですし、ビジュアル的に説明しやすい表現しやすい問題が多いと思われます。また、ゲームの AI を考えるなどの問題も身近でマンガの題材としてなじみやすいのではないでしょうか。別に厳密に有効な解法が存在しない場合でも誤魔化しやすいのではないでしょうか。

★ 終わりに

以上色々述べてきましたが、まだまだアイディアを出せる部分はあると思います。また、以上は完全に私的な見解であることを述べておきます。

誰か競技プロミングのマンガを描いて下さい。

編集後記

盛大にお休みします。

理論科学グループ　部報 第 298.5 号

2012 年 8 月 3 日　発行

発行者　河内谷耀一

編集者　久良尚任

発行所　理論科学グループ

〒 153–0041　東京都目黒区駒場 3–8–1

東京大学教養学部内学生会館 305

Telephone: 03–5454–4343

©Theoretical Science Group, University of Tokyo, 2011.
All rights reserved.

Printed in Japan.

理論科学グループ部報 第 298.5 号

— 夏合宿号 —

2012 年 8 月 3 日

THEORETICAL SCIENCE GROUP