筑波大学大学院博士課程

システム情報工学研究科修士論文

ビジュアルシステム生成系における手書き入力と図形文法との統合

山田 英仁

(コンピュータサイエンス専攻)

指導教官　田中 二郎

2003年 2月

概要

本論文では，手書き入力をビジュアルシステム生成系「恵比寿」の上で実現した「手書き入力と図形文法との統合」について述べる．本論文における手書き入力は，一般的な文字や図形の入力の他に，ジェスチャを対象としている．

恵比寿とは，図形文法に従った仕様から，ビジュアルシステムを生成する生成系である．本研究では，手書き入力に関する仕様を図形文法を用いて記述できるようにすることで，手書き入力を扱うシステムを自動的に生成できるようにする．このために，図形文法に手書き入力を示す Gesture トークンを追加することで，他の図形の記述と同様のやり方で手書き入力に関する記述を行えるようにした．また，手書きの入力の解析を行う解析器として SATIN を，手書き入力のパターンの登録に Quill を用いる．これにより，手書き入力を扱うシステムの開発者は，手書き入力のパターンを登録し，手書き入力に対する動作を図形文法で記述することで手書き入力を扱うシステムを自動的に生成することができるようになり，効率的な開発が行えるようになる．

手書き入力を扱うシステムの生成を行う生成系として「*Handragen (HANd writing DRAwing tool GENerator)*」を製作した．本論文では *Handragen* の使用例として，図形の描画および消去を実現するシステム，計算の木の描画と実行を行うシステム，および既に描かれた図形に対して整形を行うシステムを示す．

目 次

図 目 次

第1章　序論

コンピュータへの入力方式として，一般的に使用されているキーボードやマウスでの入力の他に，ペン型デバイスを用いた手書き入力方式がある．これは，ペンを用いて文字や図形を直接描くことで入力を行う方式であり，主に Personal Digital Assistant(PDA) やタブレット型コンピュータなどで用いられている．ペンを使用する方式の場合，キーボードやマウスに比べて，直観的に入力が行えるという利点がある．

手書き入力とは，一般的には文字・図形入力のことを指す．一方で，ジェスチャを用いた入力方式も存在する．本研究では，手書き入力の対象として，文字・図形入力に加え，ジェスチャも含める．

文字・図形入力とは，手書きの入力に応じた文字や図形を出力するものである．例えばペンを用いて数字の 1 の形状を手書き入力したときには画面に 1 を表示する，といったものや，ペンで丸い線を描いたときには整形された円を画面に表示する，といったものがある．

ジェスチャとは，ペンやマウスを用いた入力方式で，ペンやマウスを用いて特定の軌跡を描くことで，図形の描画や削除などの操作を行うものである．ジェスチャでは基本的に描いた軌跡は残らない．ジェスチャを使用したシステムの例として，Opera[1] などの Web ブラウザが挙げられる．Opera では，マウスの右ボタンを押しながら右から左に線を引いたとき,「戻る」という操作が実行される，といった機能がある．

手書き入力を扱うシステムを構築する場合，システムの開発者は，手書き入力のパターンの登録，手書きで入力された軌跡の取得，取得した軌跡の解析，および手書きの入力が行われたときの動作を定義する必要がある．手書き入力を扱うシステムの開発者が全ての処理を定義する場合には，開発に手間がかかる．そこで，開発の手間を軽減するシステムの構築が必要となる．

本研究では，ビジュアルシステム生成系「恵比寿」[2, 3, 4, 5, 6] の図形解析の仕組みを手書き入力の処理にも応用することで，手書き入力を扱うことのできるシステムを自動的に生成する手法を提案 [7, 8] し，それを実装したシステム「*Handragen (HANd writing DRAwing tool GENerator)*」を製作した．*Handragen* を用いることにより，手書き入力を扱うシステムの開発者は，手書き入力のパターンを登録し，手書き入力に対する動作を定義することで，手書き入力を扱うシステムを自動的に生成することができるようになる．

論文の構成は次のようになる．第 2章では準備段階として恵比寿，および SATIN について説明する．第 3章では恵比寿を図形解析の仕組みを用いた手書き入力の構築について提案する．第 4章では，第 3章で提案した手法を実装したシステム *Handragen* について述べ，第 5 章では *Handragen* を用いた手書き入力を扱うシステムの作成例について述べる．最後に第 6章で結論を述べる．

第2章　準備

2.1　ビジュアルシステム生成系「恵比寿」

2.1.1　ビジュアルシステム

ビジュアルシステムとは，ビジュアル言語を処理するシステムである．ビジュアル言語とは，文字を 1 次元に配置していくテキスト言語に対し，円や直線などの図形を 2 次元，もしくはそれ以上の次元に配置していき，意味のある図形群を構成していくものである．ビジュアル言語を用いることで，円や直線などの要素を組み合わせて状態遷移図や OMT のオブジェクト図，回路図，数式などを表現することができる．

ビジュアル言語には，テキスト言語と同様に文字や単語に相当するものがある．円や直線などの基本的な図形を図形文字と呼び，図形文字をいくつか組み合わせたものを図形単語と呼ぶ．図形単語を組み合わせて構成されるものを図形文と呼ぶ．ここで，図形文に現れるのは図形単語のインスタンスである．図形単語のインスタンスのことをトークンと呼ぶ．また，ビジュアル言語を記述する文法を図形文法と呼ぶ．

2.1.2　恵比寿

恵比寿 [2, 3, 4, 5, 6] は我々の研究室で開発しているシステムであり，ビジュアルシステムを自動的に生成することのできる生成系である (図 2.1)．恵比寿は Tcl/Tk[9] を用いて実装されている．

恵比寿では，図形文法をもとにビジュアルシステムを生成する．このため，ビジュアルシステムの開発者は，図形文法を用いて図形間の関係を記述することで，ビジュアルシステムを生成することができる．

従来，ビジュアルシステムを実装する際には，個々のシステム毎に図形の解析を行う部分を実装する必要があった．これに対し，恵比寿では図形文法を用いることで様々なビジュアルシステムを生成することができるため，新しいシステムの生成や仕様の変更といったことが効率的に行えるようになる．

2.1.3　CMG と拡張 CMG

恵比寿では，図形文法として Constraint Multiset Grammars(CMG)[10] を拡張した拡張 CMG[2, 3, 4] を用いている．

CMG とは，Marriot らが提案した図形文法である．CMG は定義したビジュアル言語の仕様を生成規則の集合として定義する．生成規則は図 2.2のように定義する．

$T(\vec{x})$ は還元される図形単語の名前 (name) である．また $T_1(\vec{x_1}), \cdots, T_n(\vec{x_n})$ は n 個の図形単

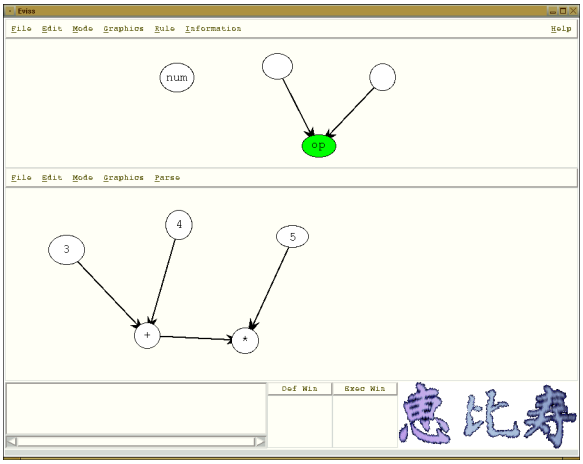

図 2.1: 恵比寿のスクリーンショット

$$
\begin{aligned}
T(\vec{x}) \quad ::= \quad & T_1(\vec{x_1}), \cdots, T_n(\vec{x_n})\ where \\
& exists\ T'_1(\vec{x'_1}), \cdots, T'_m(\vec{x'_m}) \\
& not\ exists\ T''_1(\vec{x''_1}), \cdots, T''_l(\vec{x''_l}) \\
& where\ C\ and\ \vec{x} = F.
\end{aligned}
$$

図 2.2: CMG における生成規則

$$
\begin{aligned}
T(\vec{x}) \quad ::= \quad & T_1(\vec{x_1}), \cdots, T_n(\vec{x_n})\ where \\
& exists\ T'_1(\vec{x'_1}), \cdots, T'_m(\vec{x'_m}) \\
& not\ exists\ T''_1(\vec{x''_1}), \cdots, T''_l(\vec{x''_l}) \\
& where\ C\ and\ \vec{x} = F\ and\ Action.
\end{aligned}
$$

図 2.3: 拡張 CMG における生成規則

語，もしくは図形文字であり，normal の構成要素と呼ぶ．$T'_1(\vec{x'_1}), \cdots, T'_m(\vec{x'_m})$ は m 個の図形単語，もしくは図形文字であり，exist の構成要素と呼ぶ．exist の構成要素は，図のどこかに存在する他のトークンと normal の構成要素の間に成り立つ制約を記述するために用いられる．すなわち，$T'_1(\vec{x'_1}), \cdots, T'_m(\vec{x'_m})$ が存在しなければ生成規則は適用されない．$T''_1(\vec{x''_1}), \cdots, T''_l(\vec{x''_l})$ は l 個の図形単語，もしくは図形文字であり，not_exist の生成規則と呼ぶ．$T''_1(\vec{x''_1}), \cdots, T''_l(\vec{x''_l})$ が存在するときは生成規則は適用されない．C は属性 $\vec{x_1}, \cdots, \vec{x_n}, \vec{x'_1}, \cdots, \vec{x'_m}$ に関する制約の連接である．制約とは，属性間，もしくは属性と定数の間に何らかの条件を課すことである．F は属性 $\vec{x_1}, \cdots, \vec{x_n}, \vec{x'_1}, \cdots, \vec{x'_m}$ を引数とするメソッドであり，生成される $T(\vec{x})$ の属性を定義する．C と F に not_exist の属性 $T''_1(\vec{x''_1}), \cdots, T''_l(\vec{x''_l})$ が無いのは，生成規則が適用されるためには，not_exist の構成要素があってはならないためである．

拡張 CMG では，CMG に加え，生成規則が適用されたときに図形の書き換えなどを行うアクションを定義できるようにしたものである．アクションを用いることで，ビジュアル言語の解析だけでなく，その結果を利用して何らかの処理を行うことを可能とする．

拡張 CMG の生成規則は図 2.3のようになる．CMG の生成規則に加えアクションを示す *Action* を記述することができる．アクションとして，図形の書き換えなどを行うスクリプトを記述することができる．例えば値の計算，新たな図形の生成，変更，削除などを記述する．アクションは生成規則が適用されたときに実行される．

2.1.4 拡張 CMG を用いた文法記述の例

拡張 CMG を用いた文法記述の例として，計算の木を挙げる．計算の木とは，図 2.4のようにグラフ状に数字のノードと演算子のノードをつなげたものである．ここでは，図 2.4の一番上のようなグラフを描いたとき，$(3+4)\times 5$ が計算され，図 2.4の下のように書き換えるようにする．

ここで，計算の木は次の生成規則から成る．

1. 円の中心に数字や演算子が描かれた場合，円と文字をまとめて計算の木のノードとして扱う

2. 2 つのノードから別のノードに矢印が引かれた場合には，それらの組みを 1 つのノードとして扱い，ノードが生成された場合には，計算を行い，計算結果を格納した新しいノードを生成する

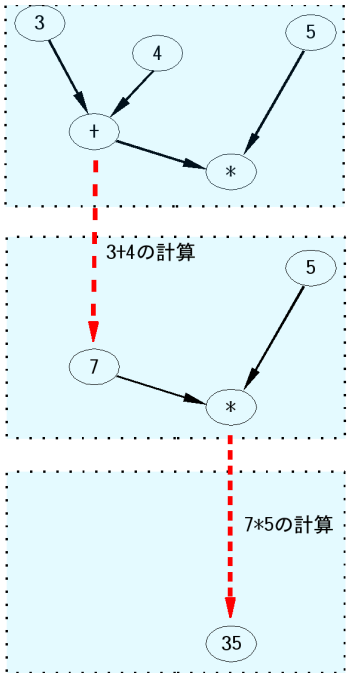

図 2.4: 計算の木を用いた $(3+4)\times 5$ の計算例

ノードを表す生成規則 node の記述・1

円の中心に数字や演算子が描かれた場合にノードとして認識する生成規則を表す node の定義は次のようになる．

```
1   node(point mid, integer value) ::= C:circle, T:text where (
2       C.mid == T.mid
3       not_exist C:Circle, L1:Line, L2:Line where (
4           C.mid == L1.end &&
5           C.mid == L2.end
6       )
7   ) {
8       mid = C.mid
9       value = to_int(T.text)
10  } {
11  }
```

上の生成規則では円 C およびテキスト T からノード N が還元される．where の後の括弧の直後には制約を記述する．制約として，円の中心座標とテキストの中心座標が同じかどうかを調べる条件式を記述する．制約の次に属性を記述する．ノードの属性には中心座標を表す mid とノードの値を表す value がある．中心座標 mid は円の中心座標と同じにする．ノードの値 value はテキストの文字を整数に変換したものを与える．ここで属性の次にはアクションを記述することができる．しかし，ここでは図形の書き換えを行わないので，アクションは記述しない．

ノードを表す生成規則 node の記述・2

2 つのノードから別のノードに矢印が引かれたときにそれらの組みを 1 つのノードとし，計算結果をもとにノードを書き換える生成規則は次のようになる．生成規則の名前は先程のものと同じ node にする．こうすることにより再帰的にノードの認識が行われるようになる．

```
1   node(point mid, integer value) ::= N1:node, N2:node, L1:line, L2:line,
2           C:circle, T:text where (
3       N1.mid == L1.start &&
4       N2.mid == L2.start &&
5       C.mid == L1.end &&
6       C.mid == L2.end &&
7       C.mid == T.mid
8   ) {
9       mid = C.mid
10      value = {script.integer {@N1.value@@T.text@@N2.value@}}
11  } {
12      delete {@N1@ @N2@ @L1@ @L2@}
13      alter @T@ text @value@
14  }
```

生成規則 node の構成要素として，別のノード N1, N2 と線 L1, L2，円 C およびテキスト T がある．制約には，線 L1 の始点がノード N1 の中心座標に，終点が円 C の中心座標に来るかどうかを調べる条件式を書く．また，線 L2 の始点がノード N2 の中心座標に，終点が円 C の中心座標に来るかという条件式も記述する．さらに，円 C の中心座標とテキスト T の中心座標が同じかどうかという条件式も記述する．ノードの属性 mid には円の中心座標を，もうひとつの属性 value には別のノードに描かれている数字どうしを，テキストに描かれている演算子で計算した結果を代入する．アクションには，計算結果をもとにノードの書き換えを行うスクリプトを記述する．ここでは，ノード N1, N2 および線 L1, L2 を消去し，テキスト T の値を計算結果に置き換えている．こうすることにより，計算された後には，計算結果を格納した 1 つのノードが出来る．

2.2 SATIN

SATIN[11] とは，Landay らによって開発された，ペンやマウスのジェスチャを扱うシステムの生成を補助するツールキットである．SATIN は Java で実装されている．SATIN を使用したアプリケーションとして，スケッチを用いて Web ページのデザインを行うことのできるシステム DENIM[12] がある (図 2.5).

SATIN にはペンやマウスの軌跡の取得を行う Interpreter と取得した軌跡の解析を行う Recognizer が存在する．SATIN ではこれらを組み合わせることによりジェスチャを扱うシステムを構築する．

システムの開発者は，ジェスチャのパターンの登録と，ジェスチャと動作の関連付けを行うことでジェスチャを扱うシステムを製作することができる．ジェスチャのパターンの登録

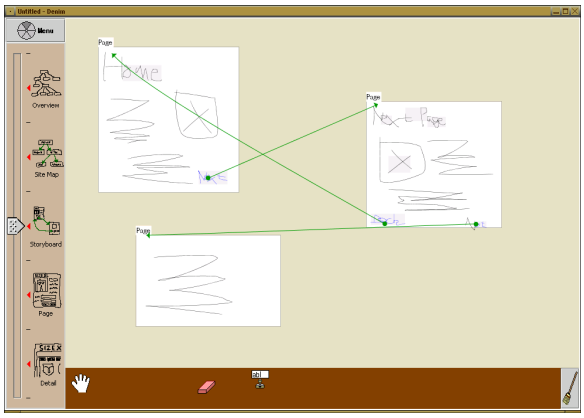

図 2.5: SATIN を使用して製作されたアプリケーション DENIM

は、Landay らが開発を行った Quill[13] を用いる．ジェスチャと動作の関連付けは，SATIN で用意されているクラスライブラリを使用し，Java で記述する．

第3章　恵比寿における手書き入力を扱うシステムの実現

3.1　手書き入力を扱うシステムの実装方法

一般的に，手書き入力を扱うシステムの実装に必要なものは，手書き入力のパターンの登録，ペンやマウスの軌跡の取得と解析，および手書き入力に対する動作の定義である．

ここで，手書き入力を扱うシステムを実装する際に全ての機能を自分で実装する方法をとった場合，機能の制限がなく，実現したいと思った機能を全て実現することが可能である．しかし，開発者は全てにおいて自分で実装する必要があり，システムの開発には時間と手間がかかる．

そこで，機能の一部を，ツールキットを使用して実現する．この場合，実現すべき機能の一部は既にツールキット側で実装されているため，実装は比較的容易である．本論文では，ツールキットとして恵比寿を用いた，手書き入力を扱うシステムの自動生成を提案する．そこで，恵比寿の拡張 CMG で手書き入力に対する動作を記述できるようにする．しかし，恵比寿にはペンやマウスで入力されたパターンを解析する機構が用意されていなため，パターンを解析するツールとして SATIN の Recognizer を使用した．また，手書き入力のパターンは Quill を用いて登録する．これにより，開発者は手書き入力に対する動作を拡張 CMG で記述することで，手書き入力を扱うシステムが恵比寿により自動生成されるため，効率的な開発が可能になる．

3.2　恵比寿における実装

3.2.1　Gesture トークンの追加

拡張 CMG で手書き入力に関する記述を行う際には，図形間の関係の記述と同じように記述できるようにすることで，図形間の関係の記述と手書き入力に関する記述をひとまとめにしてに扱えるようにしたい．こうすることで，消去および操作に関わる記述を行いやすくしたい．

恵比寿では，図形はトークン単位で表され，トークンの組み合わせを拡張 CMG で記述することにより図形間の関係を表し，ビジュアルシステムを生成している．そこで，拡張 CMG の枠組みに，手書き入力を表すトークンを新しく追加する．追加するトークンは Gesture という名前をつける．Gesture トークンに必要な属性として，パターンの判別に使用するものと，他の図形との関係を記述に使用するものの 2 種類を用意する．

Gesture トークンの属性一覧は表 3.1のようになる．

パターンの判別に使用する要素

パターンの判別に使用する要素に必要なものとして，まず手書き入力のパターン名が挙げられる．これは，拡張 CMG で処理を行う際に，パターン毎に処理を分ける必要があるためである．パターン名を参考に，パターン毎に処理を決定する．

各パターンには，ペンやマウスの軌跡が，特定のパターンに合致する確率を持たせる．パターンに応じた処理を行う際，あまりにもパターンとかけ離れている入力が来た場合に処理を行うと，誤認識の原因になる可能性がある．そこで，パターンとかけ離れている入力が来た場合には処理を行わないことが必要となってくる．パターン毎に設定されている確率の値を用いることで，入力がパターンにどれだけ合致しているかを調べることができる．

また，ペンやマウスのどのボタンが押されたか，という情報も必要となる．例えば円を描くパターンなど，描いた線を残したい場合と，円を消すパターンなど，描いた線を残したくない場合の 2 種類を別のボタンに割り当てて入力の方式を分かりやすくしたいときに用いることができる．

手書き入力と他の図形の関係の記述に使用する要素

手書き入力と他の図形との関係を記述することができるようにするため，入力されたペンやマウスの軌跡の座標値を取り，それを他の図形の座標と比較できるようにする．

具体的には次の 3 つの属性が必要になる．

1 つめはバウンディングボックスを示す図形の左上の座標や矩形の大きさである．バンディングボックスは，ペンやマウスの入力がどの図形の上で描かれたかを判別するために使用する．これを用いることで，図形のカット，コピー，ペーストなど，入力の操作対象となる図形を指定することができる．

2 つめはペンやマウスの軌跡の長さの数値である．軌跡の長さを参考にすることで，視点移動の際の視点の移動量を決める，といった操作を行うことができる．

3 つめはペンやマウスの軌跡の描き始めの座標および描き終わりの座標である．これらの座標を用いることで，グラフの 2 つの図形を結ぶ，といった操作を行いたいときに，入力が 2 つの図形にかかっているかどうかを判別することができる．

その他の要素

その他の要素として，手書き入力の時間情報が必要である．時間情報を取ることで，軌跡を入力するのに時間がかかりすぎる場合，処理を行わない，といった使用方法が考えられる．

3.2.2 Gesture トークンの例

実際にペンやマウスで軌跡を描いたとき，Gesture トークンの属性にどのような値が入るかを示す．具体例として，次の特徴を持つ軌跡を描く (図 3.1)．

- 軌跡を解析したとき，軌跡が circle パターンに合致する確率は 0.952，rectangle パターンに合致する確率は 0.715，delete パターンに合致する確率は 0.315 という結果が出た

表 3.1: Gesture トークンの属性一覧

属性	型	意味
pattern	string	手書き入力のパターン名を示す文字列とパターンに合致する確率の一覧
button	string	ペンやマウスのどのボタンが押されたか
bounds	rectangle	入力された軌跡のバウンディングボックスを表す矩形
length	integer	入力された軌跡の長さ
beginpos	point	入力された軌跡の始点の座標
endpos	point	入力された軌跡の終点の座標
time	integer	入力された軌跡の時間情報 (ミリ秒)

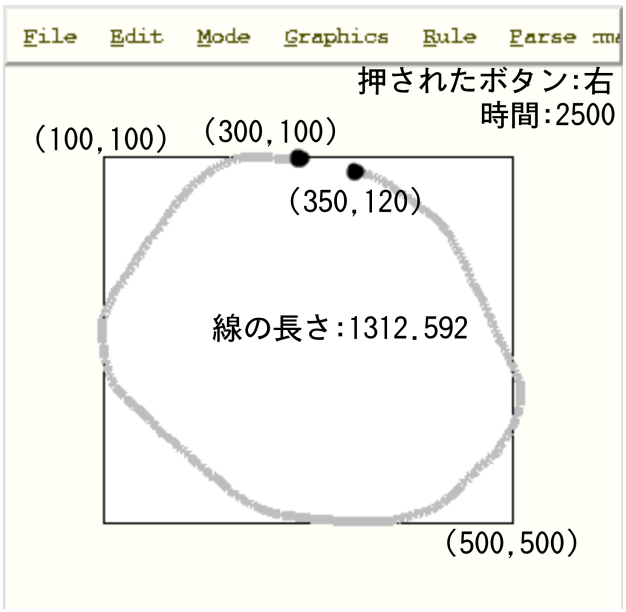

図 3.1: Gesture トークン使用例

- マウスの右ボタンドラッグにより軌跡が描かれた
- 軌跡のバウンディングボックスの左上の座標は (100, 100)，右下の座標は (500, 500) である
- 軌跡の長さは 1312.592 ピクセルである
- 軌跡の始点の座標は (300, 100) であり，終点の座標は (350,120) である
- 軌跡が描かれるのに 2500 ミリ秒かかった

このとき，Gesture トークンの各属性には表 3.2のような値が入る．

表 3.2: Gesture トークンの各属性に入る値

属性	値
pattern	circle=0.952&rectangle=0.715&delete=0.315
button	right
bounds	((100, 100), (500, 500))
length	1312.592
beginpos	(300, 100)
endpos	(350, 120)
time	2500

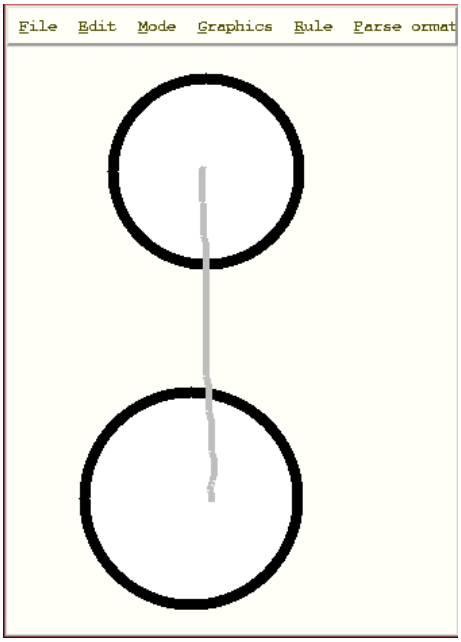

図 3.2: 円から円へ線を引くパターン

図 3.3: 円内に数字の 1 を描くパターン

3.3 類似したパターンの処理

手書き入力の軌跡を認識して処理する際に，類似したパターンに複数合致した場合どうするか，という問題がある．

類似したパターンの例として，縦に並んだ円から別の円の間に線を結ぶとき，縦の線を引く場合の入力 (図 3.2) と，円の中で数字の 1 を描くときに縦の線を引く場合の入力 (図 3.3) を考える．この 2 種類の入力は，上から下へ線を引く同じパターンであるため，手書きで入力された軌跡の意味を形のみから判別することができず，誤認識を起こす確率が高くなる．

このような，類似した軌跡を判別するために，手書き入力の形状のみを見るのではなく，入力が行われたときの状況を考慮すればよい．例えば先ほどの例の場合，次のような状況を考慮すると誤認識を起こさずに判別を行うことができる．

- ペンやマウスの軌跡が異なる 2 つの図形の上に重なっているか

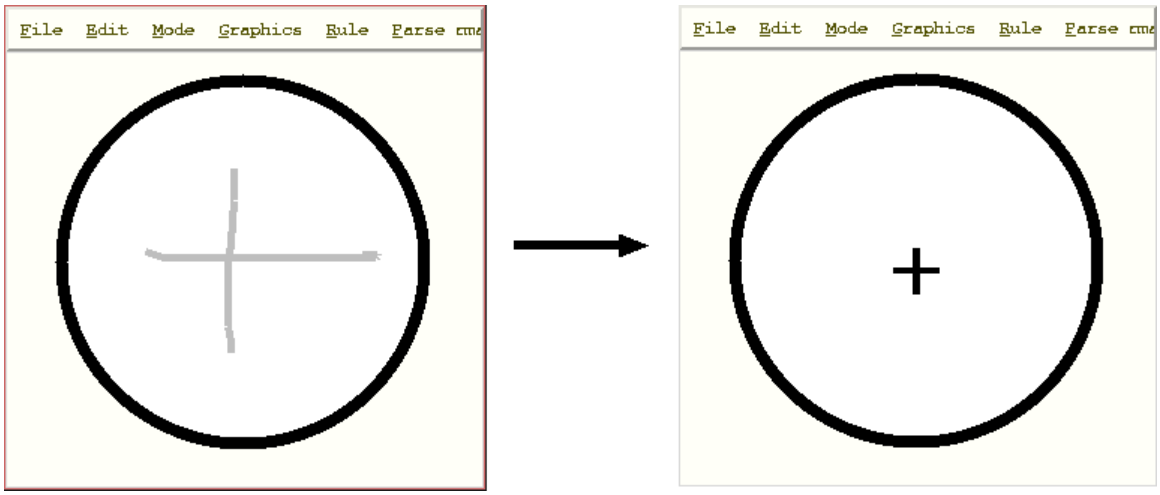

図 3.4: 手書き入力による “+” の入力

- ペンやマウスの軌跡の書き始め，または書き終わりがどの図形の上にあるか

軌跡が別の図形の上に重なっているかどうかを調べるためには，Gesture トークンのバウンディングボックスの情報である bounds 属性を利用する．具体的には，Gesture トークンの bounds 属性と，重なっているかどうか調べたい図形の座標情報を比較することでペンやマウスの軌跡が他の図形と重なっているかどうか調べることができる．

また，ペンやマウスの軌跡の書き始め，または書き終わりがどの図形の上にあるかを調べるために，Gesture トークンの beginpos 属性と endpos 属性を使用することができる．beginpos 属性および endpos 属性の座標および図形の座標を比較することでペンやマウスの軌跡が図形内にあるかどうかを調べることができる．

また，似たような軌跡を判別する方法として，字を書いているときの入力スピードと図形を描いているときの入力スピードの差を考慮に入れることで判別するという方法もある [14]．この方法を用いた場合，類似した状況のときに軌跡が描かれたときにも判別が可能である．ここでは，軌跡が描かれた時間，すなわち Gesture トークンの time 属性を利用することで判別を行う．

3.4 複数本の軌跡の組み合わせ

手書き入力の実現において，入力される軌跡は 1 つとは限らない．何本もの軌跡が入力されることが考えられる．複数の軌跡を使用する例として，演算子の “+” を描く場合がある．これは，縦線と横線の 2 本の線の組み合わせから成る (図 3.4)．

複数の軌跡が画面上に描かれた場合，1 本の線から成るパターンに合致する軌跡が複数描かれたのか，または複数の線から成るパターンが 1 つ描かれたのか，どのように判断を行うかという問題がある．判断方法として，手書き入力を行う領域をあらかじめ特定の枠の中に限定しておき，同じ枠内で描かれた軌跡をひとまとめにすることで判断する方法や，描かれた軌跡に対しバウンディングボックスを定義し，別の軌跡のバウンディングボックスと連結したり切り離したりすることで複数の軌跡から成るパターンを区別する，という方法もある

[15]．また，1 本 1 本の軌跡の位置関係をあらかじめスクリプトで用意しておくことで複数の軌跡から成るパターンを表現する方法もある [16]．

本システムでは，複数の軌跡を持つパターンについての記述も拡張 CMG で行う．本システムにおいてペンやマウスを用いて軌跡を描いたときには Gesture トークンが生成される．そこで，Gesture トークンを組み合わせることにより複数の軌跡を持つパターンを記述する．

“+” の入力を拡張 CMG で記述する場合を例にとって説明する．入力された 2 本の軌跡が “+” かどうか判定するためには，2 本の線が縦線と横線から成るかどうかを調べ，縦線と横線が交わっているかどうかを調べれば+であるかどうか分かる．2 本の線が縦線か横線かそれ以外のパターンであるかどうかは Gesture トークンの pattern 属性を調べる．2 本の線が交わっているかどうかは，それぞれの線の始点と終点の座標，すなわちそれぞれの線の beginpos 属性と endpos 属性を取得し，位置関係を調べることにより判定することができる．

ただし，複数の手書き入力パターンを扱う場合にも，3.3節で述べたように類似した軌跡が出てくる可能性があるため，状況に応じて処理を変える，といったことを行う必要がある．例えば “+” “–” の 2 つのパターンを登録してある場合，“+” を描きたいと思い縦線と横線を描いたときに，横線が “–” であると認識され，“–” が描かれ，縦線が無視されてしまう，ということが起こりうる．そこで，縦線と横線が両方描かれているか，縦線と横線が交わっているときには “+” を描き，横線のみ描かれているか，縦線と横線が交わっていないときには “–” を描く，という制約を設定することで “+” と “–” を描き分けることができる．

第4章　手書き入力と図形文法を統合したシステム「Handragen」の実装

4.1　システム構成

4.1.1　従来の恵比寿における処理の流れ

従来の恵比寿におけるシステム構成は図 4.1のようになる．

従来の恵比寿は，メニューから図形を選択し，マウスを用いて図形の入力を行う図形入出力部，生成規則の入力を行う生成規則入力部，入力された図形を生成規則に従って解析するパーサから構成される．

ユーザは，まず生成規則入力部で生成規則を入力する．次に，図形入出力部で線分や円などの図形を入力する．図形が入力されると，図形の形や座標などのデータがパーサに送信される．パーサでは，あらかじめ入力された生成規則をもとに入力された図形の位置関係などを解析し，生成規則に応じた図形の書き換えなどを行う．解析の結果は，即座に図形入出力部に反映される．

4.1.2　*Handragen* における処理の流れ

恵比寿で手書き入力を取得して処理を行わせるために，手書き入力を拡張 CMG で扱えるようにする．そのために，まずペンやマウスの軌跡を取得して，図形文法に従って解析できるようにする必要がある．そこで，従来は図形描画の際の範囲指定にのみ使用していたペンやマウスの軌跡を保存する．ペンやマウスの軌跡を取得した後は軌跡に対し解析を行い，解析結果をパーサ送信する．

提案するシステムの構成は図 4.2のようになる．

本システムにおける処理の流れは次のようになる．まず，図形入出力部で，ペンやマウスを用いて図形や手書きの入力を行う．図形入出力部で描かれた図形はそのままパーサへ送られる．一方，ペンやマウスの軌跡のデータはパターン解析部に送られ，解析が行われる．解析が行われた後，解析の結果がパーサに送られる．パーサでは，図形および手書き入力の解析結果を，あらかじめ生成規則入力部で入力された生成規則に基づいて解析を行う．パーサで解析された結果は，図形入出力部に反映され，図形の書き換えなどが行われる．

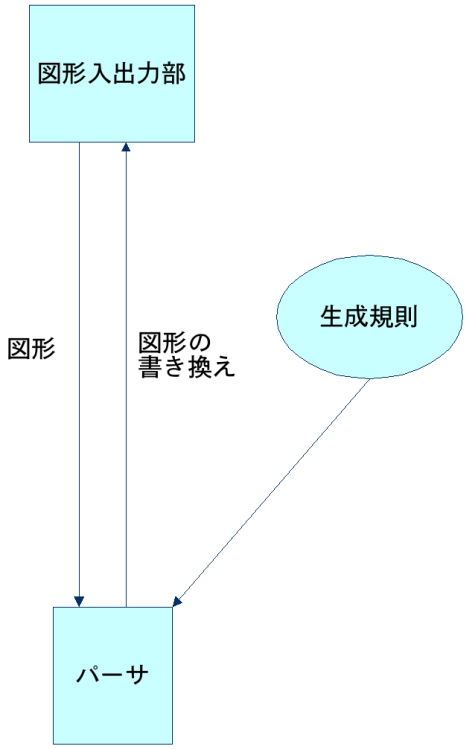

図 4.1: 従来の恵比寿のシステム構成図

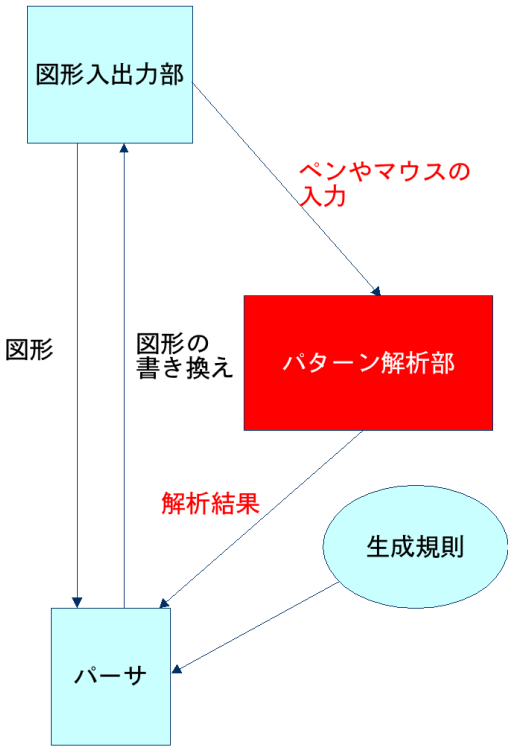

図 4.2: *Handragen* の構成図

4.2 各部分の実装

4.2.1 ペンやマウスの軌跡の取得

ペンやマウスの軌跡の取得は図形入出力部で行う．軌跡の取得を開始するタイミングは，ペンの場合は画面やタブレットにペンが触れたときで，マウスの場合はボタンが押されたというイベントを取得したときである．また，軌跡の取得を終了するタイミングは，ペンの場合は画面やタブレットにペンが離れたときで，マウスの場合はボタンが離されたというイベントを取得したときである．

軌跡として保存するデータとして，ペンやマウスがたどった全ての座標を対象とする．軌跡の取得を開始した時点でペンやマウスの現在の座標の記録を行う．ペンやマウスがドラッグされたときは，そのつど座標を一覧に追加する．

ペンが画面から離れたとき，またはマウスのボタンが離されたときには，座標の取得を終了して，座標の一覧をパターン解析器へ送信する．

4.2.2 パターンの認識

パターンの認識には SATIN の Recognizer を用いる．ただし，*Handragen* の主な部分は Tcl/Tk で書かれているが，SATIN は Java で書かれているため，*Handragen* と SATIN に互いにデータを渡すインタフェース部が必要となる．ここで，インタフェース部を実装する言語として Java を選択した．なぜなら，SATIN も Java で実装されているため，SATIN の Recognizer のメソッドを直接呼び出すことができるからである．また，*Handragen* の主な部分とインタフェース部のデータのやりとりはソケット通信を用いて実装した．

図形入出力部からインタフェース部へ送信するデータは，ペンやマウスの入力データである．ペンやマウスの入力データは，ペンやマウスの軌跡を表す．ペンやマウスの軌跡は，一定間隔で取得したペンやマウスの座標の集合として表される．ペンやマウスの座標の集合は，以下のフォーマットを用いて表現する．

$x_1, y_1 \llcorner x_2, y_2 \llcorner x_3, y_3 \cdots x_n, y_n$

これは，軌跡の集合が n 個の座標から成る場合のフォーマットである．ペンやマウスの座標の集合は文字列で表される．1 個 1 個の座標は x 座標, y 座標 というようにカンマで区切られる．座標の組はスペースで区切られ 1 つの文字列となる．この文字列を SATIN に送信する．

図形入出力部から送られてきたペンやマウスの入力データは，インタフェース部で処理される．最初に送信されてきた文字列をスペースで分割する．これにより，座標を 1 つずつ処理することができる．スペースで分割した文字列をさらに “,” で区切ることにより，x 座標と y 座標に分けることができる．文字列の分割は StringTokenizer クラスを用いる．このようにして分割した座標を SATIN の Recognizer に渡す．

SATIN の Recognizer で解析した後は，解析結果をインタフェース部からパーサへ送信する．以下の要素を解析結果とする．

- パターン名の候補
- 入力された軌跡の解析結果が，上のパターン名である確率

全ての候補に対して上のようなデータを生成する．

さらに，以下のペンやマウスの入力に関する以下のデータも生成して送信を行う．

- バウンディングボックス
- 入力した軌跡の長さ
- 入力を開始した点の座標
- 入力を終了した点の座標

インタフェース部から送られてきた結果をもとにパーサが Gesture トークンを生成し，空間解析を行う．

4.2.3 パターンを記したファイルの複数読み込み

手書き入力のパターンはファイルに記録されており，インタフェース部でファイルを適宜読み込んで使用する．パターンを記録したファイルは，システムに応じて複数作ることができる．

ここで，ファイルを複数読み込む場合も想定される．しかし，SATIN では複数のファイルを読み込むことができない．そこで，複数のファイルを読み込むことができるように拡張を行う．

SATIN では，ファイルの読み込みおよびペンやマウスの入力の解析は RubineRecognizer クラスで行っている．RubineRecognizer クラスでは以下の手順でファイルの読み込みを行っている．

- ファイル名をもとに FileReader クラスのインスタンスを生成する
- FileReader クラスのインスタンスを引数に GestureSet クラスのインスタンスを生成，それをもとに RubineClassifier クラスのインスタンスを生成

また，RubineRecognizer クラスでは以下の手順でペンやマウスの入力の解析を行う．

- ペンやマウスの軌跡の情報をもとに Gesture クラスのインスタンスを生成する
- RubineClassifier クラスの classify_satin メソッドを呼び出し解析を行う
- 解析結果を，(パターン名, 確率) の組みにして返す

これを，複数のファイルに対応させるために，以下の拡張を行った．

- 読み込むファイルの数だけ RubineClassifier クラスのインスタンスを生成
- ペンやマウスの入力の解析を行うとき，RubineClassifier クラスの全てのインスタンスに対し classify_satin メソッドを呼び出す
- 全ての解析結果をまとめ，(パターン名, 確率) の組みにして返す

4.2.4 Gesture トークンの生成

ペンやマウスの軌跡の解析が終了し，解析結果がパーサへ返された後に Gesture トークンを生成する．解析結果をもとに，Gesture トークンの属性に値を入れていく．Gesture トークンを生成した後は，他のトークンと同様に，トークン一覧を表すトークンデータベースに登録する．

複数の軌跡を描いた場合は，描いた軌跡の本数だけ Gesture トークンが生成される．ただし，複数の Gesture トークンがある場合，空間解析を行うタイミングを考える必要がある．なぜなら，1 つの Gesture トークンが生成されるごとに空間解析を行うと，少ない Gesture トークンを使用した生成規則が優先されることになり，誤動作の原因につながる．このため，複数本の軌跡を考慮した空間解析を行うことが困難になる．

そこで，軌跡が描かれてから 1 秒経ったときに空間解析を行う．

4.2.5 Gesture トークンの消去

生成された Gesture トークンはトークンデータベースに登録され，様々な操作に使用される．しかし，全ての Gesture トークンが使用されるわけではなく，使用されない Gesture トークンも存在する．ここで，使用されなかった Gesture トークンはずっとトークンデータベースに残る．この状態で解析を行ったとき，使われなかったはずの Gesture トークンが解析に使われてしまい，誤動作を起こす可能性がある．

例えば，円の中に数字を表すパターンを描いたときのみ円の中に数字が描かれ，その他の場所に数字を表すパターンを描いても何も起こらない，という状況を設定する．このとき，制約条件として円内に数字を表す Gesture トークンがあった場合にのみ円を描く，というように設定する．この条件の下では，円の外に数字を表すパターンを描いたとき，円外に Gesture トークンがずっと残る結果となる．その後でこの Gesture トークン上に円を描いたとき,「円内に数字を表すパターンを描いた」という条件が成立してしまい，数字を表すパターンを描いていないのに円内に数字が描かれてしまう，ということが起こる．

これを防止するため，使用されなかった Gesture トークンは定期的に消去する必要がある．ここでは，Gesture トークンに設定してある時間情報を調べ，生成されてから 1 秒以上経過しており，かつ使用されていない Gesture トークンを消去する．

第5章　手書き入力を扱うシステムの作成例

5.1　図形の描画と消去

基本的な手書き入力の例として，図形の描画と消去について述べる．以下の仕様に基づいて図形の描画や消去を行う生成規則を定義する．

- 丸形の線を描いたとき，整形された円が描かれる
- 円の上でギザギザの線を描いたとき，円が消去される

このとき，軌跡の描きかたによっては，軌跡が円の描画を示すパターンと円の消去を示すパターンの両方に合致してしまい，誤動作を起こす可能性がある．そこで，該当するパターン名の確率に対ししきい値を設け，さらに最も高い確率を持つパターン名を調べることで，誤動作を防止する．制約は次のように定める．

- 登録されている全てのパターンに対する確率を取得し，一番確率の高いパターンが delete だった場合，確率によらず円を消去する
- 一番確率の高いパターンが delete ではなく，かつ circle パターンの確率が 0.6 以上だった場合，円を描画する

以上の方針をもとに，円を描画する生成規則 CreateCircle および円を消去する生成規則 DeleteCircle を定義する．

円を描画する生成規則 CreateCircle の記述

CreateCircle の記述は次のようになる．制約部分では，まずパターンが circle で，かつ確率が 0.6 以上であるかどうかを調べる．パターンと確率の検索には search_gesture というメソッドを用いている．これは，引数として与えたパターン名がもう一つの引数として与えた数値以上の確率だった場合に True を返す．次に一番高いパターン名が delete でないことを調べる．これには not_highest_gesture メソッドを用いる．

制約が満たされたとき，アクションが実行される．アクション部分では，create メソッドを用いて円を生成する．円の座標は Gesture トークンの bounds 属性の座標から求める．最後に，Gesture トークンを消去する．

```
1   C:CreateCircle() ::= G:Gesture where (
2       search_gesture("circle", prob >= 0.6) &&
3       not_highest_gesture("delete")
4   ) {} (
```

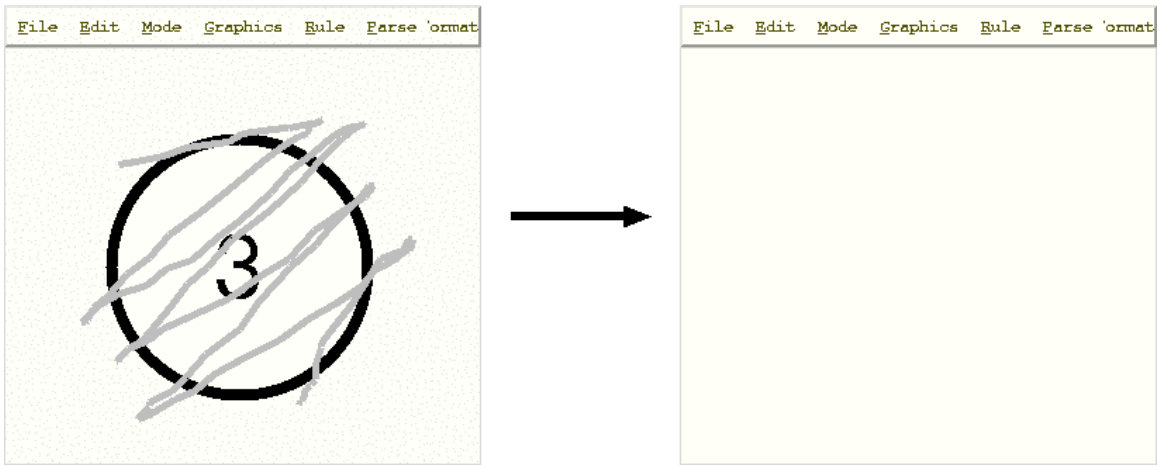

図 5.1: 手書き入力を用いた円の消去

```
5       create circle @G.bounds_lu_x@ @G.boudns_lu_y@ @G.bounds_rl_x@ \
6   @G.bounds_rl_y@
7       delete {@G@}
8   )
```

円を消去する生成規則 DeleteCircle の記述

DeleteCircle の記述は次のようになる． 制約部分では highest_gesture メソッドを用いて最も確率の高いパターン名が “delete” になっているかを調べる．また，Circle トークンの中心点・半径および Gesture トークンのバウンディングボックス bounds をもとに，円と軌跡が重なっているかどうかを調べ，円を消去するかどうかを決定する．アクション部分では，Circle トークンの消去を行う．

```
1   D:DeleteCircle() ::= G:Gesture, C:Circle where (
2       highest_gesture("delete") &&
3       touch(C, G.pattern)
4   ) {} (
5       delete {@C@ @G@}
6   )
```

上記の図形文法を恵比寿上で入力したあとでマウスの右ボタンを押しながら円の上でギザギザの線を描くと，円が消去される (図 5.1).

5.2 計算の木

手書き入力を用いて 2.1.4節で述べた計算の木を描画する．計算の木においてはノード，エッジおよび中の数字や演算子を手書き入力を用いて記述できるようにする．手書き入力を用いた計算の木の記述においては，先に挙げた円の描画・消去を行う生成規則に加え，次の生成規則を追加する．

- 円の上で数字や演算子を描いたとき，描いた軌跡に合致するパターンに応じた数字や演算子を円内に描く
- 円の上で数字や演算子を描いたとき，既に円内に数字や演算子が描かれていた場合，文字を追加する
- 円ともう一つの円を結ぶように線を引いたとき，始点から終点に向かって矢印が引かれる

また，誤動作を防ぐため，以下の制約を追加する．

- 数字や演算子が描かれたとき，それが円内で描かれたかどうかを調べ，円内で描かれたときのみ数字や演算子を描く
- ペンやマウスの軌跡の始点が円内にあり，軌跡の終点が別の円内にあるときに限り，矢印を引く

以上の方針をもとに，先ほどの例で定義した円の描画・消去を行う生成規則に加え，円を描く生成規則 CreateCircle を変更し，数字や演算子を描く生成規則 DrawString および矢印を描く生成規則 CreateEdge を追加する．

円を描く生成規則 CreateCircle の変更

CreateCircle では，円内で円のパターンに合致する軌跡を描いたときに円が描画されないようにするため，ペンやマウスの軌跡の始点と終点が円の外にあるかどうかを調べる，という条件を追加する．ここでは not_exist 制約を用いて条件を記述している．

```
1  C:CreateCircle() ::= G:Gesture where (
2      search_gesture("circle", prob >= 0.6) &&
3      not_highest_gesture("delete")
4      not_exist N:Circle where (
5          in_circle(N, G.beginpos) &&
6          in_circle(N, G.endpos)
7      )
8  ) {} (
9      create circle @G.bounds_lu_x@ @G.bounds_lu_y@ @G.bounds_rl_x@ \
10 @G.bounds_rl_y@
11     delete {@G@}
12 )
```

数字や演算子を描く生成規則 DrawString の記述

DrawString では，ペンやマウスの軌跡の始点と終点が円内にあるかどうか，という条件を in_circle メソッドを用いて記述する．どの数字や演算子が描かれたという情報は search_string メソッドを使用して取得する．この情報をもとにテキストを円の中心に生成する．

```
D:DrawString() ::= G:Gesture C:Circle where (
    in_circle(C, G.beginpos) &&
    in_circle(C, G.endpos) &&
    not_highest_gesture("delete")
) {} (
    set string [search_string]
    create text @C.mid_x@ @C.mid_y@ -text $string
    delete {@G@}
)
```

数字や演算子を追加する生成規則 AppendString の記述

AppendString では，ペンやマウスの軌跡の始点と終点が既に数字や演算子が記述された円内にあるかどうか，という条件を記述する．アクションでは，まずノードの値を変更してから，ノードに描かれているテキストを変更する．古いテキストに新しいテキストを追加する形でテキストの内容を決定する．テキストの変更には alter メソッドを用いる．

```
A:AppendString() ::= G:Gesture, N:Node where (
    in_circle(C, G.beginpos) &&
    in_circle(C, G.endpos) &&
    not_highest_gesture("delete")
) {} (
    set val @N.value@
    set string [search_string]
    alter @N.value@ $val$string
    alter [lindex @N.children@ 1] text @N.value@
    delete {@G@}
)
```

矢印を描く生成規則 DrawEdge の記述

DrawEdge では，まず軌跡のパターンが edge に合致する確率が 0.6 以上かどうか調べる条件を記述する．また，ペンやマウスの軌跡の始点が片方の円内に，終点がもう片方の円内にあるかどうかを調べる，という条件を exist 制約を用いて記述する．

```
C:CreateEdge() ::= G:Gesture where (
    search_gesture("edge", prob >= 0.6) &&
    not_highest_gesture("delete")
    exist N1:Circle, N2:circle where (
        in_circle(N1, G.beginpos) &&
        in_circle(N2, G.endpos)
    )
) {} (
```

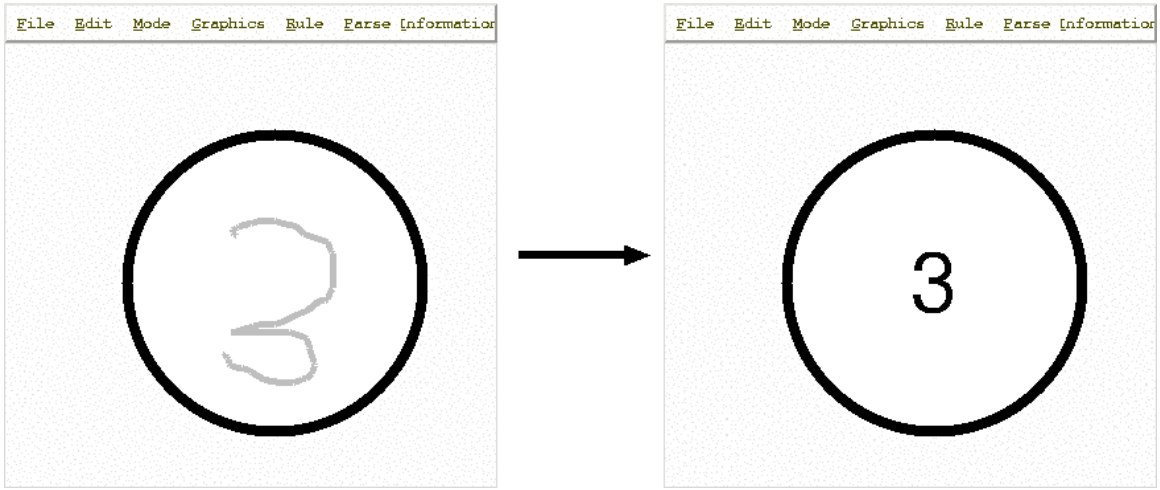

図 5.2: 手書き入力を用いた数字や演算子の描画

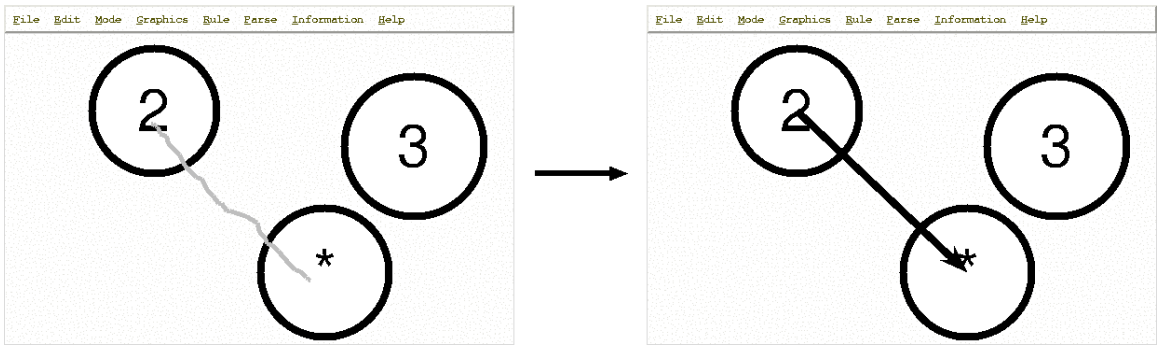

図 5.3: 手書き入力を用いた矢印の描画

```
 9        create line @N1.mid_x@ @N1.mid_y@ @N2.mid_x@ @N2.mid_y@ -arrow last
10        delete {@G@}
11    )
```

これにより，円の中で手書きで数字や演算子を描くことにより円内に文字が描画される (図 5.2)．また，円同士を直線で結ぶことにより，矢印を引くことができる (図 5.3)．

5.3 既に描かれた図形に対する操作

Handragen を使用することで，既に描かれた図形に対し，手書き入力を用いた変更を記述することが可能となる．ここでは，五十嵐らの Pegasus[17] に見られる，2 本の線分を平行にする例と，2 本の線分の長さを等しくする例を記述する．

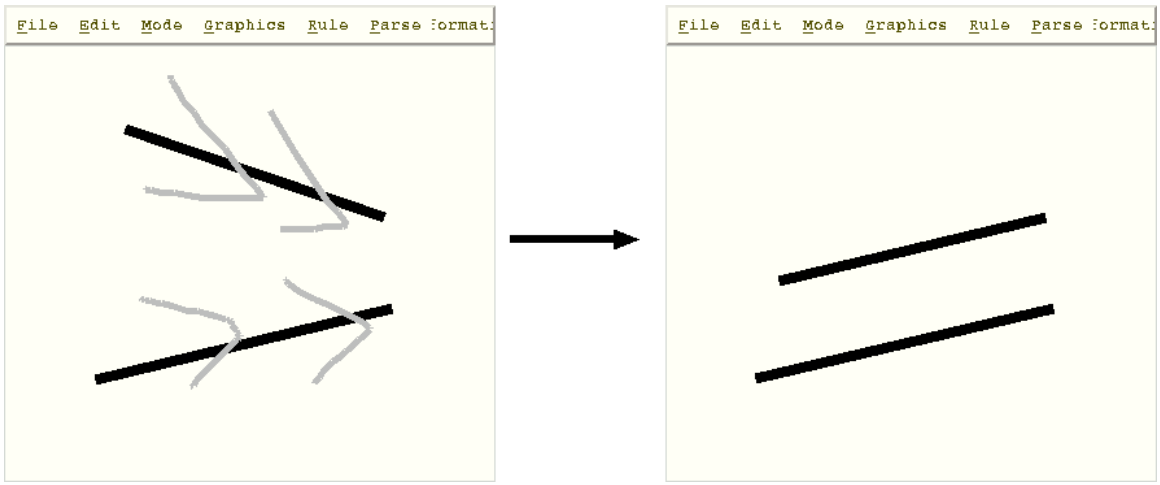

図 5.4: 手書き入力を用いて 2 本の線分を平行にする例

5.3.1 2 本の線分を平行にする例

2 本の線分に対し,「く」の字型の軌跡を 2 本ずつ，それぞれの線分の上に描くことで，2 本の線分を平行にするような手書き入力を定義する (図 5.4).

ペンやマウスでの手書き入力があったときに，2 本の線分を平行にするための制約は次のようになる.

- 「く」の字をした軌跡が描かれた (これ以降,「く」の字のパターンを parallel パターンとする)
- parallel パターンに合致する軌跡が，それぞれの線分の上に 2 つずつ描かれた
- 各々の線分の上にある parallel パターンに合致する軌跡が十分に近い

描かれた軌跡が parallel パターンに合致するかどうかは，Gesture トークンの pattern 属性を見ることで調べることができる.

線分の上に parallel パターンに合致する軌跡が描かれたかどうかについては，線分が軌跡のバウンディングボックスに交わるかどうかで判定を行う．このために，Line トークンの属性 start と end，および Gesture トークンの bounds 属性を調べる．ここで，線分がバウンディングボックスの上を通るかどうか調べるメソッド intersect_rect_and_line を用いた.

各々の線分の上にある軌跡が十分に近いかどうかは，2 つの軌跡のバウンディングボックスが重なっているかどうかで判定を行う．そのために，Gesture トークンの bounds 属性を調べる．ここで，バウンディングボックス同士が重なるかを調べるメソッド touch を用いた.

さらに，制約を満たしたとき，2 つの線分を平行にする必要がある．ここで，片方の線分がもう一方の線分と平行になるような座標を求めるメソッド parallel_lines を用いる．線分の座標の書き換えはメソッド alter を用いている.

2 つの線分を平行にする手書き入力を拡張 CMG で表すと次のようになる.

```
1   ParallelLine() ::= G1:Gesture, G2:Gesture, G3:Gesture, G4:Gesture,
2           L1:Line, L2:Line where (
```

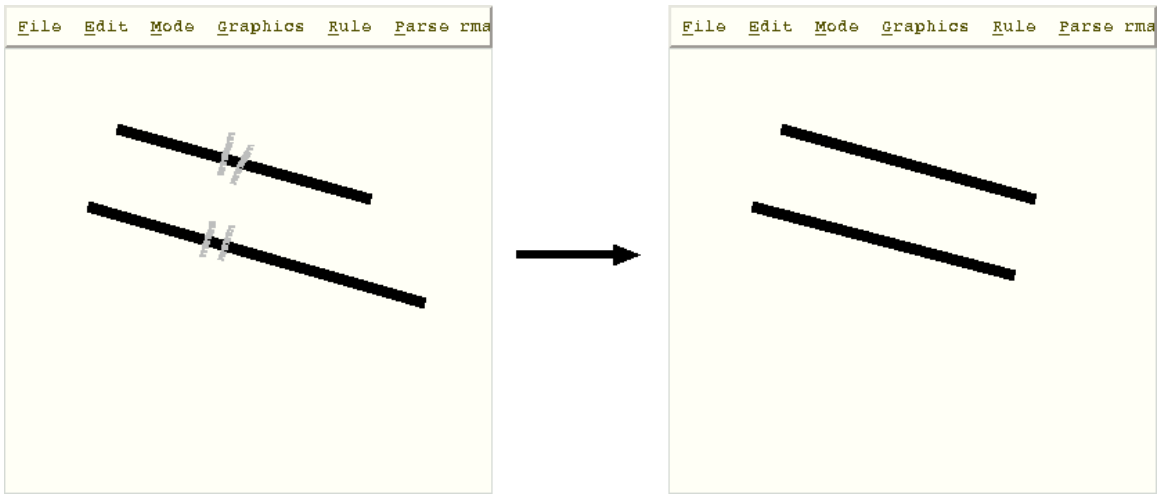

図 5.5: 手書き入力を用いて 2 本の線分を等しくする例

```
3       search_gesture(G1, "parallel", prob >= 0.5) &&
4       search_gesture(G2, "parallel", prob >= 0.5) &&
5       intersect_rect_and_line(G1.bounds, L1) &&
6       intersect_rect_and_line(G2.bounds, L1) &&
7       intersect_rect_and_line(G3.bounds, L2) &&
8       intersect_rect_and_line(G4.bounds, L2)
9       touch(G1, G2) &&
10      touch(G3, G4)
11  ) {} {
12      set parallel_axis [parallel_lines(@L1@, @L2@)]
13      alter @L2@ start_x [lindex parallel_axis 0]
14      alter @L2@ start_y [lindex parallel_axis 1]
15      delete {@G1@ @G2@}
16  }
```

5.3.2 2 本の線分の長さを等しくする例

2 本の線分に対し，短い，まっすぐの軌跡を 2 本ずつ，それぞれの線分の上に描くことで，2 本の線分の長さを等しくするような手書き入力を定義する (図 5.5).

ペンやマウスでの手書き入力があったときに，2 本の線分の長さを等しくするための制約は次のようになる.

- 短いまっすぐの軌跡が描かれた (これ以降，このパターンを equals パターンとする)
- equals パターンに合致する軌跡が，それぞれの線分の上に 2 つずつ描かれた
- 各々の線分の上にある equals パターンに合致する軌跡が十分に近い

描かれた軌跡が equals パターンに合致するかどうかは，Gesture トークンの pattern 属性を見る．

equals パターンに合致する軌跡が，それぞれの線分の上に 2 つずつ描かれたかどうかを調べる制約，および各々の線分の上にある軌跡が十分に近いかどうかを調べる制約は，線分を平行にするときの制約と同様に，Line トークンの start，end 属性および Gesture トークンの bounds 属性を使用する．線分を平行にするときと同様に，メソッド intersect_rect_and_line とメソッド touch を用いる．

さらに，制約を満たしたとき，2 つの線分の長さを等しくするためのメソッド equal_lines を用いる．線分の座標の書き換えはメソッド alter を用いている．

2 つの線分を平行にする手書き入力を拡張 CMG で表すと次のようになる．

```
1   EqualLine() ::= G1:Gesture, G2:Gesture, G3:Gesture, G4:Gesture,
2           L1:Line, L2:Line where (
3       search_gesture(G1, "equals", prob >= 0.5) &&
4       search_gesture(G2, "equals", prob >= 0.5) &&
5       intersect_rect_and_line(G1.bounds, L1) &&
6       intersect_rect_and_line(G2.bounds, L1) &&
7       intersect_rect_and_line(G3.bounds, L2) &&
8       intersect_rect_and_line(G4.bounds, L2)
9       touch(G1, G2) &&
10      touch(G3, G4)
11  ) {} {
12      set equals_axis [equal_lines(@L1@, @L2@)]
13      alter @L2@ start_x [lindex equals_axis 0]
14      alter @L2@ start_y [lindex equals_axis 1]
15      delete {@G1@ @G2@}
16  }
```

第6章　まとめ

本研究ではペンを用いた文字や図形の入力およびペンやマウスによるジェスチャを手書き入力の重要な要素として認識した．その上で，文字や図形の入力およびジェスチャを拡張 CMG で記述できるようにすることで，ビジュアルシステム生成系「恵比寿」を用いた手書き入力を扱うシステムの自動生成を行えるようにした．また，従来の恵比寿では手書き入力を処理する機構が無かったため，SATIN を用いた手書き入力の解析を行えるようにするためのインタフェース部分の製作を行った．これにより，手書き入力を扱うシステムの開発者は，手書き入力のパターンを登録し，手書き入力に対する動作を記述するのみでシステムの開発が行えるようになり，開発を効率的に行うことができる．

さらに，手書き入力と図形文法を統合したシステム「*Handragen*」を開発し，実際に *Handragen* を用いた手書き入力を扱うシステムの開発例を示した．

謝辞

本論文を進めるにあたり，終始ご指導下さった指導教官の田中二郎教授，および志築文太郎助手には心から感謝いたします．

また，IPLab の皆様からは，貴重なご意見を頂きました．特に，飯塚和久さん，亀山裕亮さん，Visual System グループの皆様からは研究および実装の方針について大変貴重なご意見を頂きました．ここに感謝の意を表します．

参考文献

[1] Opera web site. `http://www.opera.com/`.

[2] 馬場昭宏, 田中二郎. Spatial parser generator を持ったビジュアルシステム. 情報処理学会論文誌, Vol.39, No.5, pp. 1385–1394, 1998.

[3] 馬場昭宏, 田中二郎. 「恵比寿」を用いたビジュアルシステムの作成. 情報処理学会論文誌, Vol.40, No.2, pp. 497–506, 1999.

[4] Akihiro Baba and Jiro Tanaka. Eviss: A visual system having a spatial parser generator. In *Proceedings of Asia Pacific Computer Human Interaction 1998 (APCHI'98), IEEE Computer Society Press*, pp. 158–164, 1998.

[5] 馬場昭宏, 田中二郎. GUI を記述するためのビジュアル言語. インタラクティブシステムとソフトウェア V, 日本ソフトウェア科学会 WISS'97, pp. 135–140, 1997.

[6] 馬場昭宏, 田中二郎. Spatial parser generator の tcl/tk を用いた実装. インタラクション'97 論文集, pp. 71–78, 1997.

[7] 山田英仁, 飯塚和久, 田中二郎. ビジュアルシステム生成系「恵比寿」におけるジェスチャの実現. 日本ソフトウェア科学会第 19 回大会, 2002. CD-ROM.

[8] 山田英仁, 飯塚和久, 田中二郎. ビジュアルシステム生成系「恵比寿」におけるジェスチャの実現. 第 10 回 インタラクティブシステムとソフトウェアに関するワークショップ (WISS2002), pp. 155–156, 2002.

[9] Tcl developer site. `http://www.tcl.tk/`.

[10] Richard Helm, Kim Marriott, and Martin Odersky. Building visual language parsers. In *Conference proceeding on Human Factors in Computer Systems (CHI '91)*, pp. 105–112, 1991.

[11] Jason I. Hong and James A. Landay. SATIN: A toolkit for informal ink-based applications. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST2000)*, pp. 63–72, 2000.

[12] James Lin, Mark W. Newman, Jason I. Hong, , and James A. Landay. DENIM: Finding a tighter fit between tools and practice for web site design. In *Conference proceeding on Human Factors in Computer Systems (CHI 2000)*, pp. 510–517, 2000.

[13] A. Chris Long Jr., James A. Landay, Lawrence A. Rowe, and Joseph Michiels. Visual similarity of pen gestures. In *Conference proceeding on Human Factors in Computer Systems (CHI 2000)*, pp. 360–367, 2000.

[14] 豊原純平, 佐賀総人. ファジィスプライン曲線の分割処理に基づく書描弁別法の提案. インタラクティブシステムとソフトウェア VIII, 日本ソフトウェア科学会 WISS2000, pp. 13–18, 2000.

[15] Elizabeth D. Mynatt, Takeo Igarashi, W. Keith Edwards, and Anthony LaMarca. Flatland: New dimensions in office whiteboards. In *Conference proceeding on Human Factors in Computer Systems (CHI '99)*, pp. 346–353, 1999.

[16] Mark D. Gross. Recognizing and interpreting diagrams in design. In *Proceedings of Advanced Visual Interfaces '94 (AVI '94)*, pp. 88–94, 1994.

[17] Takeo Igarashi, Satoshi Matsuoka, Sachiko Kawachiya, and Hidehiko Tanaka. Interactive beautification: A technique for rapid geometric design. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'97)*, pp. 14–17, 1997.

付 録A　Handragenの使用手引き

Handragen を用いた手書き入力を扱うシステムの製作方法について述べる．製作するシステムとして，丸形の軌跡を描いたら円を描画するものを例に挙げる．パターン名は circle として登録を行う．

A.1　手書き入力パターンの登録

手書き入力パターンの登録は Quill を用いて行う．Quill を起動するには，*Handragen* のメニューから「Rule」→「Resist the Gesture」を選択する．Quill の画面構成は図 A.1のようになる．

Quill は 4 つの要素から成る．左上にあるのが，登録されたパターンの一覧を示す Training Set/Test Sets エリア，左中央にあるのが手書き入力によりパターンを登録する Drawing エリア，右にあるのがパターン毎に描かれた軌跡の一覧を表示する Window エリア，下にあるのがログが表示される Log エリアである．

Quill で新しいパターンを登録するには，まず Quill のメニューから「Gesture」→「New Gesture Category」を選択する．そうすると，Training Set/Test Sets エリアに新しいパターンが出来る．パターン名を右クリックすることにより，名前の変更を行うことができる．ここで，パターンの名前を「circle」に変更する．また，パターン名を左ダブルクリックすることにより Window エリアに新しいウィンドウが表示される．

パターンの登録は Drawing エリアで行う．Drawing エリアで軌跡を描くごとに Window エリアに新しい軌跡が追加される．

パターン名を決定してパターンを登録するという作業を，登録したいパターンの数だけ行う．ここでは circle パターン 1 つのみの登録なので，これ以上パターンは追加しない．

目的のパターンを登録したら，Quill のメニューから「File」→「Save」を選択し，パターンをファイルに記録する．

A.2　手書き入力に対する動作の記述

手書き入力に対する動作の記述は，*Handragen* 上で行う．生成規則を入力する画面を開くために，恵比寿のメニューから「Rule」→「Make New Production Rule」を選択する．そうすると，生成規則を入力するためのウィンドウが立ち上がる．

生成規則入力ウィンドウは，生成規則の名前を入力する欄，属性を入力する欄，アクションを入力する欄，制約を入力する欄，そして構成要素を入力する欄から構成される (図 A.2).

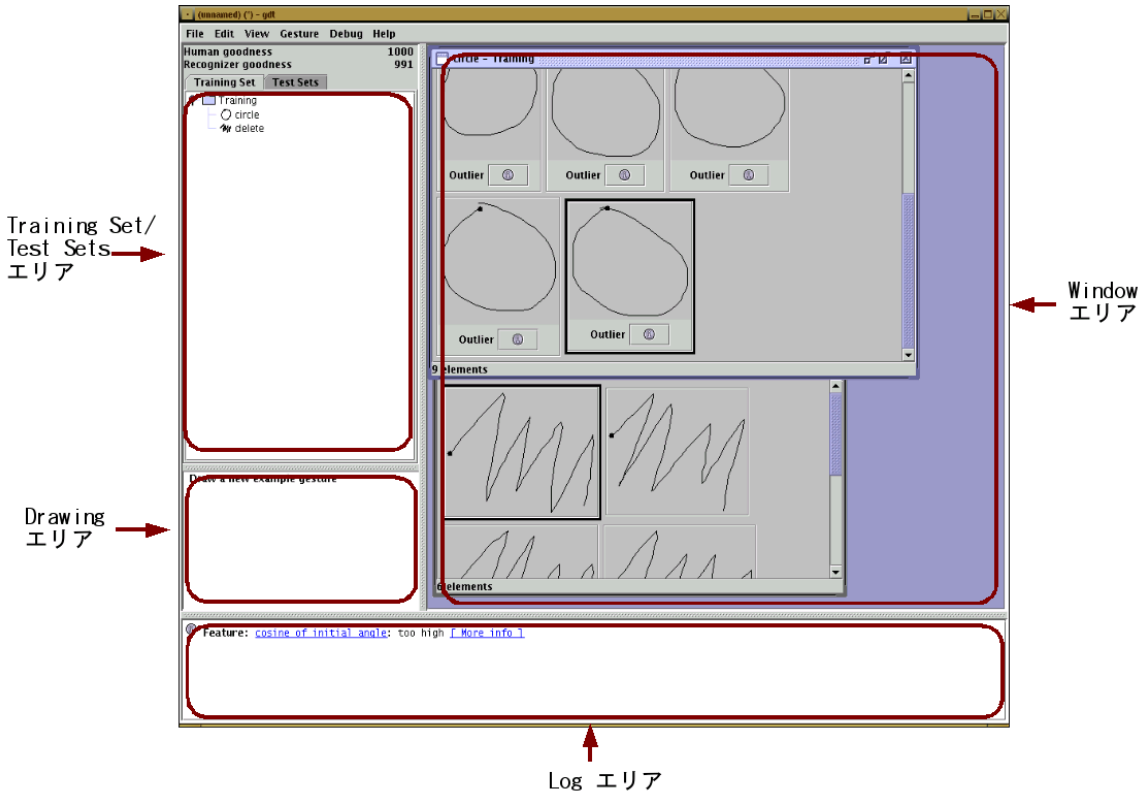

図 A.1: Quill を用いたパターン登録の画面

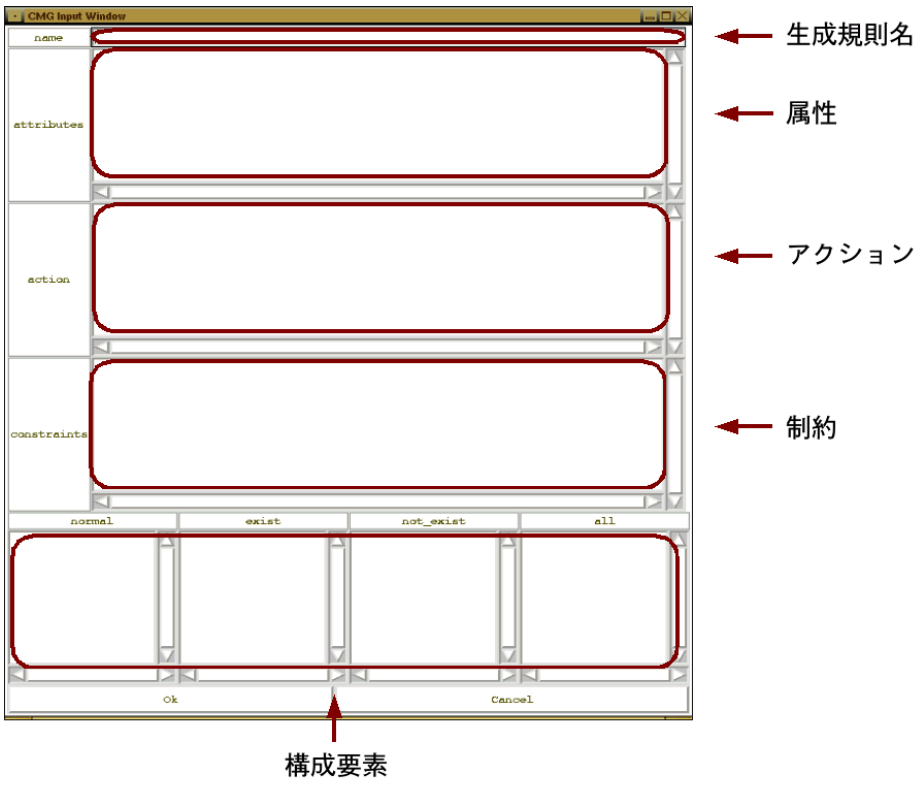

図 A.2: 生成規則入力ウィンドウ

ここでは，以下の生成規則を入力する．

```
1   C:CreateCircle() ::= G:Gesture where (
2       search_gesture("circle", prob >= 0.6)
3   ) {} (
4       create circle @G.bounds_lu_x@ @G.boudns_lu_y@ @G.bounds_rl_x@ \
5   @G.bounds_rl_y@
6       delete {@G@}
7   )
```

まず，生成規則の名前を入力する欄に

```
CreateCircle
```

と入力する．

次に，属性を入力する欄は，属性が定義されていないので空欄にする．

アクションを入力する欄には，

```
create circle @0.0.bounds_lu_x@ @0.0.boudns_lu_y@ @0.0.bounds_rl_x@ \
@0.0.bounds_rl_y@
delete {@0.0@}
```

と入力する．

制約を入力する欄には，

```
search_gesture {string circle} {double 0.6}
```

と入力する．

最後に，構成要素を入力する 3 つの欄のうち，左の「normal」と書かれている欄に

```
gesture
```

と入力する．

全ての生成規則を入力した状態の生成規則入力ウィンドウは図 A.3のようになる．

入力した生成規則を保存したいときには，*Handragen* のメニューから「Rule」→「Save Production Rules」を選択する．メニューを選択するとファイル保存ダイアログが開くので，適当なファイル名を入力する．

また，保存した生成規則を読み込みたいときにはメニューから「Rule」→「Load Production Rules」を選択する．メニューを選択するとファイル読み込みダイアログが開くので，読み込みたいファイル名を選択する．

A.3 製作したシステムの使用

最後に，製作したシステムを実際に使用する方法について述べる．

まず，生成規則が入力されているか確かめる．これは，*Handragen* のメニューから「Rule」→「Show Rule List」を選択することで別ウィンドウ上にルール一覧が出てくるので，先程登録した「CreateCircle」があるかどうか確かめる．

次に，circle パターンを記録したファイルを読み込む．*Handragen* のメニューから「Rule」→「Add Pattern File」を選択する．メニューを選択するとファイル読み込みダイアログが開くので，circle パターンを記録したファイルを指定する．

この状態で，ペンやマウスの左ドラッグを用いて丸形の軌跡を描くと，円が生成される．

図 A.3: 生成規則 CreateCirle を入力したときの画面

付 録B　システムのソースコード

Handragen のソースコードのうち，手書き入力に関わる部分を抜粋する．

B.1　Tcl/Tk を使用した部分

主に恵比寿を拡張した部分に対しては，Tcl/Tk を用いて実装を行った．

B.1.1　canvas_procs.tcl

```
1   proc buttonpressed {x y} {
2       global prevx prevy
3       global x_list y_list
4
5       set prevx $x
6       set prevy $y
7       set x_list $x
8       set y_list $y
9   }
10
11  proc buttondragged {canvas x y} {
12      global prevx prevy
13      global x_list y_list
14
15      # prev point to current point
16      $canvas create line $prevx $prevy $x $y -width 5 -fill gray -tags
17  gesture
18
19      set prevx $x
20      set prevy $y
21      lappend x_list $x
22      lappend y_list $y
23  }
24
25  proc buttonreleased {canvas button x y} {
26      global x_list y_list
```

```
27      global HostName Port
28      global D
29      global NumOfTokens
30
31      set len [llength $x_list]
32
33      set str ""
34
35      for {set i 1} {$i < $len} {incr i} {
36          set xx [lindex $x_list $i]
37          set yy [lindex $y_list $i]
38
39          append str "$xx,$yy "
40      }
41
42      # Create socket and send gesture to SATIN
43      set sock [socket $HostName $Port]
44      puts $sock $str
45      flush $sock
46
47      # Rename stroke tags
48      $canvas itemconfigure gesture -tags token_id$NumOfTokens
49
50      # Create Gesture token
51      foreach item [make_gesture_token_list [gets $sock] $button] {
52          add $item
53      }
54  #    incremental_parser
55
56      close $sock
57  }
58
59  ##########################################################################
60  # bindings for Define window and Test window
61
62  proc keybind {penmode} {
63      if {$penmode} {
64          set normalbutton 3
65          set gesturebutton 1
66      } else {
67          set normalbutton 1
```

```
68          set gesturebutton 3
69      }
70
71      foreach j {.c .test} {
72          # select items
73          $j bind all <Button-$normalbutton> "set curXcn \[set curX %x\];
74  set curYcn \[set curY %y\];select_item $j \[$j find withtag current\]"
75
76          # select items incrementally
77          $j bind all <Shift-Button-$normalbutton> "set curXcn \[set curX %x\];
78  set curYcn \[set curY %y\];select_item_inc $j \[$j find withtag current\]"
79
80          # select text
81          $j bind text <Button-$normalbutton> "change_text $j \
82  [$j find withtag current\] \%x \%y"
83
84          # create items
85          bind $j <Button-$normalbutton> "start_create_item $j \%x \%y"
86          bind $j <Button$normalbutton-Motion> ""
87          bind $j <ButtonRelease-$normalbutton> ""
88
89          # move items selected
90          $j bind selected <Button$normalbutton-Motion> "move_item_constr5 $j
91  \%x \%y"
92
93
94          # move items selected
95          $j bind selected <ButtonRelease-$normalbutton>
96  "move_item_constr_and_parse $j \%x \%y"
97
98
99          # delete selection incrementally
100         bind $j <Shift-Button-$gesturebutton> "delete_selection_inc $j"
101     }
102
103     bind .test <Button-2> {
104         buttonpressed %x %y
105     }
106
107     bind .test <Button2-Motion> {
108         buttondragged .test %x %y
```

```
109       }
110
111       bind .test <ButtonRelease-2> {
112           buttonreleased .test "middle" %x %y
113       }
114
115       bind .test <Button-$gesturebutton> {
116           global isgesture
117
118           buttonpressed %x %y
119           set isgesture false
120       }
121
122       bind .test <Button$gesturebutton-Motion> {
123           global isgesture
124
125           buttondragged .test %x %y
126           if {$isgesture == "false"} {
127               set isgesture true
128           }
129       }
130
131       # popup mode menu or delete selection
132       bind .c <ButtonRelease-$gesturebutton> {
133           if {[.c find withtag selected] != {}} {
134               delete_selection .c
135           } else {
136               tk_popup .mbarc.mode.menu [winfo pointerx .] [winfo pointery .]
137           }
138       }
139       bind .test <ButtonRelease-$gesturebutton> {
140           global isgesture
141
142           if {$isgesture == "true"} {
143               buttonreleased .test "right" %x %y
144           } else {
145               if {[.test find withtag selected] != {}} {
146                   delete_selection .test
147               } else {
148                   tk_popup .mbartest.mode.menu [winfo pointerx .]
149   [winfo pointery .]
```

```
150             }
151         }
152     }
153
154     # clear system messages
155     bind .ext.mes.x.mes <Double-Button-$normalbutton> {clear_sys_messages}
156 }
```

B.1.2 rule.tcl

```
1   proc make_gesture_token_list {pat button} {
2       global NumOfTokens StartTime GesturePatternList
3       set id $NumOfTokens
4       incr id
5
6       set token_list []
7       set tmp_list []
8       set CurTime [clock clicks -milliseconds]
9
10      lappend tmp_list {string type gesture}
11
12      lappend tmp_list [list string mousebutton $button]
13
14      set attlist [split $pat ?]
15
16      foreach i $attlist {
17          set valuelist [split $i ,]
18
19          if {[lindex $valuelist 1] == "result"}
20  { # If the list expresses gesture
21              # Set all pattern
22              set pat_prob [lindex $valuelist 2]
23              set GesturePatternList [split $pat_prob &]
24          }
25          lappend tmp_list $valuelist
26      }
27
28      if {[winfo exists .gesturelist]} {
29          .gesturelist.t delete 0.0 end
30          foreach i $GesturePatternList {
31              set ii [split $i =]
```

```
32              set pat [string range "[lindex $ii 0]                " 0 9]
33              set prb [lindex $ii 1]
34              .gesturelist.t insert end "$pat$prb\n"
35
36          }
37      }
38
39      lappend tmp_list [list integer time [expr $CurTime-$StartTime]]
40
41      lappend token_list $tmp_list
42
43      return $token_list
44  }
```

B.1.3 items.tcl

```
1   proc delete_gesture_token {time canvas} {
2       global D NumOfTokens ReallyDeleted StartTime
3
4       set curtime [expr [clock clicks -milliseconds]-$StartTime]
5
6       # At least more than or equal 1 gesture, call incremental parser
7       set isgesture 0
8       for {set i 0} {$i < $NumOfTokens} {incr i} {
9           if {[lsearch $ReallyDeleted $i] == -1} {
10              if {[sb_var value $D($i.type)] == "gesture"} {
11                  set eventtime [sb_var value $D($i.time)]
12                  if {[expr $curtime-$time*2] > $eventtime} {
13                      set isgesture 1
14                  }
15              }
16          }
17      }
18      if {$isgesture == 1} {
19           incremental_parser
20      }
21
22      # Delete other gesture token
23      for {set i 0} {$i < $NumOfTokens} {incr i} {
24          if {[lsearch $ReallyDeleted $i] == -1} {
25              if {[sb_var value $D($i.type)] == "gesture"} {
```

```
26                  set eventtime [sb_var value $D($i.time)]
27                  if {[expr $curtime-$time*2] > $eventtime} {
28                      delete_token $i 1
29                  }
30              }
31          }
32      }
33
34      after $time delete_gesture_token $time $canvas
35  }
```

B.2 Javaを使用した部分

主にSATINとのインタフェース部にあたる箇所は，Javaを用いて実装を行った.

B.2.1 SatinRecognizerServer.java

```
1   import java.net.*;
2   import java.io.*;
3   import java.util.*;
4   import java.text.*;
5   import java.awt.geom.*;
6
7   import edu.berkeley.guir.lib.awt.geom.*;
8   import edu.berkeley.guir.lib.debugging.*;
9   import edu.berkeley.guir.lib.gesture.*;
10  import edu.berkeley.guir.lib.satin.*;
11  import edu.berkeley.guir.lib.satin.event.*;
12  import edu.berkeley.guir.lib.satin.objects.*;
13  import edu.berkeley.guir.lib.satin.recognizer.*;
14  import edu.berkeley.guir.lib.satin.recognizer.rubine.*;
15  import edu.berkeley.guir.lib.satin.stroke.*;
16  import edu.berkeley.guir.lib.satin.interpreter.*;
17
18  /**
19   *    SATINのRecognizer渡すデータを受け取るためのサーバ
20   *    Usage: java SatinRecognizerServer (port)
21   *    デフォルトのポート番号:10005
22   *
23   *    ChangeLog
24   *    2003/1/30 パターンファイルを外部から読み込めるようにした
```

```
25     *     2003/1/21 パターンの候補に dot を含めないようにした
26     *     2002/11/14 graffiti.gsa のバグに対応 確率の小数点 4 桁以下は切るようにした
27     *     2002/10/4 複数の結果を Tcl 側に送るようにした
28     *     2002/9/30 複数のファイルを読み込めるようにした
29     *     2002/9/8  ストロークの情報 (バウンディングボックスなど) を恵比寿側に送信するようにした
30     *               また、確率を 0.8 に下げた
31     *     2002/8/19 Recognizer が返す認識結果の確率を考慮　確率が 0.9 以上のとき、認識に成功したとみなすようにした
32     *     2002/7/1? 最初のバージョン
33     */
34    public class SatinRecognizerServer implements SatinConstants {
35        private MultiFileRecognizer rr;
36
37        private static final Debug debug = new Debug(Debug.ON);
38        private static final WeakHashMap cache = new WeakHashMap();
39        private int port;
40
41        public SatinRecognizerServer(int p) {
42            port = p;
43
44            // Recognizer の初期化
45            rr = new MultiFileRecognizer();
46            // rr = initializeRecognizer("pattern/graphmouse.gsa");
47            // addFile(rr, "pattern/graffiti.gsa");
48
49            //  rr = initializeRecognizer("pattern/graffiti.gsa");
50            // addFile(rr, "pattern/graphmouse.gsa");
51            // addFile(rr, "pattern/calcmouse.gsa");
52            // addFile(rr, "pattern/standard.gsa");
53
54            // rr = initializeRecognizer("pattern/geometry.gsa");
55        }
56
57        public void recognizeStart() {
58            ServerSocket svSocket=null;
59            BufferedReader is;
60            PrintWriter os;
61
62            TimedStroke tstk = new TimedStroke();
```

```
63          Classification c;
64          String result, recognizeresult;
65          double prob;
66
67          try{
68              //ソケットを作成します。
69              svSocket = new ServerSocket(port);
70          }catch(IOException e){
71              System.err.println( e.getMessage() );
72              System.exit(1);
73          }
74
75          Socket cliSocket = null;
76          try{
77              //クライアントからのコネクション要求を受け付けます。
78              cliSocket = svSocket.accept();
79          }catch( IOException e ){
80              System.err.println( e.getMessage() );
81              System.exit(1); }
82          try{
83              //入力ストリームを作成します。
84              is = new BufferedReader( new InputStreamReader
85  ( cliSocket.getInputStream()));
86
87              //出力ストリームを作成します。
88              os = new PrintWriter( new OutputStreamWriter
89  ( cliSocket.getOutputStream()));
90
91              //クライアントから送信された文字列を受信します。
92              String a = is.readLine();
93              final String patternstr = new String("patternfile");
94
95              // System.out.println("/" + a + "/");
96              // 受信した文字列の解析
97              if (a.equals("exit")) {
98                  System.exit(0);
99              } else if (a.equals("OK")) {
100                 cliSocket.close();
101                 svSocket.close();
102                 return;
103             } else if (a.startsWith(patternstr)) {
```

```
104                     String patfilename = a.substring(patternstr.length() + 1);
105                     addFile(rr, patfilename);
106                     cliSocket.close();
107                     svSocket.close();
108                     return;
109                 }
110
111                 StringTokenizer st = new StringTokenizer(a);
112                 while (st.hasMoreTokens()) {
113                     StringTokenizer cst = new StringTokenizer(st.nextToken(), ",");
114                     double dx, dy;
115
116                     dx = Double.parseDouble(cst.nextToken());
117                     dy = Double.parseDouble(cst.nextToken());
118                     // System.out.println("x = " + dx + " y = " + dy);
119                     tstk.addPoint(dx, dy);
120                 }
121
122                 //ジェスチャ解析結果を送信します。
123                 c = rr.classify(tstk);
124                 prob = Double.parseDouble(c.getFirstValue().toString()); // 確
    率の取得
125
126                 // 確率が 0.8 以上だったら認識に成功したとみなす
127                 if (prob >= 0.1) {
128                     recognizeresult = (String)c.getFirstKey();
129                     recognizeresult = recognizeresult.trim();
130                 }
131                 else {
132                     recognizeresult = "Failed analization";
133                 }
134
135                 printResult(c);
136                 result = analizeStroke(tstk, c);
137
138                 os.println(result);
139                 os.flush();
140                 //ストリーム，ソケットをクローズします。
141                 is.close();
142                 os.close();
143                 cliSocket.close();
```

```
144             svSocket.close();
145         }catch( IOException e ){
146             e.printStackTrace();
147         }
148     }
149 
150     public static void main( String[] args ){
151         SatinRecognizerServer srs;
152         int tmpport;
153 
154         // ポート番号の設定
155         // 引数から取得
156         tmpport = 10005;
157         if (args.length == 1) {
158             tmpport = Integer.parseInt(args[0]);
159         }
160 
161         srs = new SatinRecognizerServer(tmpport);
162 
163         while (true) {
164             srs.recognizeStart();
165         }
166     }
167 
168     private MultiFileRecognizer initializeRecognizer(String str) {
169         MultiFileRecognizer recognizer = null;
170 
171         System.out.println("Initializing Recognizer...");
172 
173         //// 0. Check the cache first.
174         recognizer = (MultiFileRecognizer) cache.get(str);
175         if (recognizer != null) {
176             System.out.println("Done.");
177             return (recognizer);
178         }
179 
180         //// 1. Boundary case.
181         if (str.equals("")) {
182             recognizer = new MultiFileRecognizer();
183             cache.put(str, recognizer);
184             System.out.println("Done.");
```

```
185             return (recognizer);
186         }
187
188         //// 2. Try to retrieve from disk.
189         try {
190             String          strFileName = str;
191
192             FileInputStream fistream    = new FileInputStream(strFileName);
193             Reader          rdr         = new InputStreamReader(fistream);
194
195             //// 2.1. Create the MultiFileRecognizer.
196             recognizer = new MultiFileRecognizer(rdr);
197             cache.put(str, recognizer);
198             System.out.println("Done.");
199             return (recognizer);
200         }
201         catch (Exception e) {
202             debug.println(e);
203         }
204
205         //// 3. Otherwise, retrieve from the network.
206         try {
207             //// 3.1. Get the properties for where Rubine's data is.
208             String      strURL;
209             URL         url;
210             InputStream istream;
211             Reader      rdr;
212
213             strURL = str;
214
215             url    = new URL(strURL);
216
217             //// 3.2. Start up a Reader opening the file with
218             //// the Rubine data set.
219             istream = url.openStream();
220             rdr     = new InputStreamReader(istream);
221
222             //// 3.3. Create the MultiFileRecognizer.
223             recognizer = new MultiFileRecognizer(rdr);
224             cache.put(str, recognizer);
225             System.out.println("Done.");
```

```
226             return (recognizer);
227         }
228         catch (FileNotFoundException e) {
229             debug.println(e);
230             debug.println("Initialization Error - using empty
231 Rubine Recognizer");
232             recognizer = new MultiFileRecognizer();
233         }
234         catch (ParseException e) {
235             debug.println(e);
236             debug.println("Bad Rubine data file.");
237             debug.println("Initialization Error - using empty
238 Rubine Recognizer");
239             recognizer = new MultiFileRecognizer();
240         }
241         catch (TrainingException e) {
242             debug.println(e);
243             debug.println("Error in Rubine Recognizer processing data file.");
244             debug.println("Initialization Error - using empty
245 Rubine Recognizer");
246             recognizer = new MultiFileRecognizer();
247         }
248         catch (IOException e) {
249             debug.println(e);
250             debug.println("Initialization Error - using empty
251 Rubine Recognizer");
252             recognizer = new MultiFileRecognizer();
253         }
254 
255         System.out.println("Done.");
256         return (recognizer);
257     }
258 
259     private void addFile(MultiFileRecognizer rr, String str) {
260         System.out.println("Adding File " + str + "...");
261 
262         //// 0. Check the cache first.
263         //// 1. Boundary case.
264         if (str.equals("")) {
265             System.out.println("Done.");
266             return;
```

```
267         }
268
269         //// 2. Try to retrieve from disk.
270         try {
271             String          strFileName = str;
272
273             FileInputStream fistream    = new FileInputStream(strFileName);
274             Reader          rdr         = new InputStreamReader(fistream);
275
276             //// 2.1. Create the MultiFileRecognizer.
277             rr.addFile(rdr);
278             cache.put(str, rr);
279             System.out.println("Done.");
280         }
281         catch (Exception e) {
282             debug.println(e);
283         }
284
285         //// 3. Otherwise, retrieve from the network.
286         try {
287             //// 3.1. Get the properties for where Rubine's data is.
288             String      strURL;
289             URL         url;
290             InputStream istream;
291             Reader      rdr;
292
293             strURL = str;
294
295             url    = new URL(strURL);
296
297             //// 3.2. Start up a Reader opening the file with
298             //// the Rubine data set.
299             istream = url.openStream();
300             rdr     = new InputStreamReader(istream);
301
302             //// 3.3. Create the MultiFileRecognizer.
303             rr.addFile(rdr);
304             cache.put(str, rr);
305             System.out.println("Done.");
306         }
307         catch (FileNotFoundException e) {
```

```
308             debug.println(e);
309             debug.println("Initialization Error - using empty
310 Rubine Recognizer");
311         }
312         catch (ParseException e) {
313             debug.println(e);
314             debug.println("Bad Rubine data file.");
315             debug.println("Initialization Error - using empty
316 Rubine Recognizer");
317         }
318         catch (TrainingException e) {
319             debug.println(e);
320             debug.println("Error in Rubine Recognizer processing data file.");
321             debug.println("Initialization Error - using empty
322 Rubine Recognizer");
323         }
324         catch (IOException e) {
325             debug.println(e);
326             debug.println("Initialization Error - using empty
327 Rubine Recognizer");
328         }
329
330         System.out.println("Done.");
331     }
332
333     /**
334      *    Recognizerからの認識結果を基に、恵比寿側へ返す結果の文字列を作成する
335      */
336     public String analizeStroke(TimedStroke stk, Classification cls) {
337         Rectangle2D bounds = stk.getBounds2D();
338         int boundrl_x = (int)(bounds.getX() + bounds.getWidth()),
339             boundrl_y = (int)(bounds.getY() + bounds.getHeight());
340         int length = (int)stk.getLength2D(COORD_ABS);
341         Point2D beginpos = stk.getStartPoint2D(COORD_ABS);
342         Point2D endpos = stk.getEndPoint2D(COORD_ABS);
343
344         // 認識結果の一覧を作成する
345         StringBuffer sbstr = new StringBuffer();
346         boolean isFirst = true;
347         DecimalFormat dec = new DecimalFormat();
348         dec.setMaximumFractionDigits(3);
```

```
349
350             /*
351              * まずマップ cls の最初のキーと値を取り出す
352              * その後、取ったキーを消去する
353              * その後で最初のキーを取り出すメソッドを使うと、次のキーを取り出せるという寸法
354              * ソートされたリスト一覧が得られないという SATIN の仕様のため、このようなトリッキーなことをしています　ご了承ください
355              *
356              * あとマップの値 (確率) については小数点 4 桁以下は切ってあります (送信量削減のため)
357              */
358             while (!cls.isEmpty()) {
359                 String key;
360
361                 if (!isFirst) {
362                     sbstr.append("&");
363                 }
364                 key = cls.getFirstKey().toString();
365                 if (key.equals("dot")) {
366                     cls.remove(cls.getFirstKey());
367                     continue;
368                 }
369                 sbstr.append(key);
370                 sbstr.append("=");
371                 sbstr.append(dec.format(cls.getFirstValue()));
372                 cls.remove(cls.getFirstKey());
373                 isFirst = false;
374             }
375
376             return new String("string,result," + sbstr.toString() +
377                               "?point,lu," + bounds.getX() + " " + bounds.getY() +
378                               "?integer,lu_x," + bounds.getX() +
379                               "?integer,lu_y," + bounds.getY() +
380                               "?point,rl," + boundrl_x + " " + boundrl_y +
381                               "?integer,rl_x," + boundrl_x +
382                               "?integer,rl_y," + boundrl_y +
383                               "?integer,length," + length +
384                               "?point,beginpos," + beginpos.getX()
385                               + " " + beginpos.getY() +
386                               "?integer,beginpos_x," + beginpos.getX() +
```

```
387                             "?integer,beginpos_y," + beginpos.getY() +
388                             "?point,endpos," + endpos.getX()
389                             + " " + endpos.getY() +
390                             "?integer,endpos_x," + endpos.getX() +
391                             "?integer,endpos_y," + endpos.getY());
392         }
393
394         /**
395          *    解析結果を表示する (デバッグ用)
396          */
397         public void printResult(Classification classification) {
398             System.out.println(classification);
399             System.out.println();
400         }
401     }
```

B.2.2 MultiFileClassifier.java

```
1   //// See bottom of source code for software license
2
3   import java.util.*;
4   import edu.berkeley.guir.lib.gesture.*;
5   import edu.berkeley.guir.lib.gesture.util.*;
6   import java.io.PrintStream;
7
8   import edu.berkeley.guir.lib.collection.*;
9
10  public class MultiFileClassifier
11     extends Classifier {
12
13     //============================================================================
14     //===   CONSTANTS   ==========================================================
15
16     private static final double EPSILON = 1E-6;   // for singular matrix check
17     public  static final String DOT     = "dot";  // the name of a dot stroke
18
19     //===   CONSTANTS   ==========================================================
20     //============================================================================
21
22
23
```

```
24      //=========================================================================
25      //===   NONLOCAL VARIABLES   ==============================================
26
27      boolean flagTrained = false;
28
29      //===   NONLOCAL VARIABLES   ==============================================
30      //=========================================================================
31
32
33
34
35      //=========================================================================
36      //===   CONSTRUCTORS   ====================================================
37
38      public MultiFileClassifier() {
39         super(null);
40      }
41
42      //-------------------------------------------------------------
43
44      /**
45       * Does not automatically train.
46       */
47      public MultiFileClassifier(GestureSet gs) {
48         super(gs);
49      }
50
51      //===   CONSTRUCTORS   ====================================================
52      //=========================================================================
53
54
55
56      //=========================================================================
57      //===   CLASSIFICATION METHODS   ==========================================
58
59      /**
60       * An internal version of classify for use with Satin.
61       * I know that copying and pasting code is generally a bad practice, but
62       * it will take a significant amount of work to unify and separate the two.
63       */
64      public SortedValueNumMap classify_satin(Gesture gesture)
```

```
65          throws Exception {
66
67          //// 1. Make sure recognizer is trained.
68          if (flagTrained == false) {
69             flagTrained = true;
70             train();
71          }
72
73          //// This map contains the name of objects recognized
74          //// as well as the raw values. The values will be scaled
75          //// to percentages by the RubineRecognizer wrapper.
76          SortedValueNumMap map = new SortedValueNumMap();
77          map.setAscending(false);
78
79          //// 2. Not trained, no data, so fail.
80          if ((gestureSet == null) || (gestureSet.size() == 0)) {
81             return null;
82          }
83
84          List enabledCategories = gestureSet.getEnabledCategories();
85
86          //// 3. Check for dot first.
87          if (gesture.size() == 1) {
88             if (dotCategory == null) {
89                map.put(DOT, 1.0);
90                return (map);
91             }
92             for (Iterator iter = enabledCategories.iterator(); iter.hasNext();) {
93                GestureCategory gestureCategory = (GestureCategory) iter.next();
94                if (gestureCategory == dotCategory)
95                   map.put(gestureCategory.getName(), 1.0);
96                   return (map);
97             }
98             map.put(DOT, 1.0);
99             return (map);
100         }
101
102         //// 4. Compute discriminant.  Higher disc -> higher probability that
103         ////    it's the right category.
104         int    numCategories = enabledCategories.size();
105         double disc[]        = new double[numCategories];
```

```
106
107         for (int catIndex = 0; catIndex < numCategories; catIndex++) {
108            double sum = weights[catIndex][0];
109            for (int featureNum = 0; featureNum < featureClasses.length;
110                                                                  featureNum++) {
111               Feature feature =
112                  FeatureFactory.getFeature(featureClasses[featureNum], gesture);
113               sum += weights[catIndex][featureNum+1] * feature.getValue();
114            }
115            disc[catIndex] = sum;
116         }
117
118         int    bestGC = -1;
119         int    i      = 0;
120         double denom;
121
122         //// Calculate accuracies.
123         FeatureVector fv = new FeatureVector(gesture);
124         for (Iterator iter = enabledCategories.iterator(); iter.hasNext(); i++) {
125            GestureCategory category = (GestureCategory) iter.next();
126            if (!category.isDot()) {
127               denom = calculateAccuracy(disc, i);
128
129               map.put(category.getName(),
130                       MahalanobisDistance(fv.getValues(), meanFeatureValues[i]));
131            }
132         }
133
134         return map;
135
136      } // of method
137
138
139
140
141
142      //-----------------------------------------------------------------
143
144      /**
145       * @param disc          is the array of dicriminants.
146       * @param j             is the entry in disc whose accuracy we want
```

```
147       */
148      protected double calculateAccuracy(double[] disc, int j) {
149         double denom = 0;
150         for (int i = 0; i < disc.length; i++) {
151            // quick check to avoid computing negligible term
152            double d = disc[i] - disc[j];
153            if (d > -7.0) {
154               denom += Math.exp(d);
155            }
156         }
157         return (denom);
158      } // of method
159
160      //===   CLASSIFICATION METHODS   ==========================================
161      //=========================================================================
162
163   }
164
165   //=============================================================================
166
167   /*
168   Copyright (c) 2000 Regents of the University of California.
169   All rights reserved.
170
171   Redistribution and use in source and binary forms, with or without
172   modification, are permitted provided that the following conditions
173   are met:
174
175   1. Redistributions of source code must retain the above copyright
176      notice, this list of conditions and the following disclaimer.
177
178   2. Redistributions in binary form must reproduce the above copyright
179      notice, this list of conditions and the following disclaimer in the
180      documentation and/or other materials provided with the distribution.
181
182   3. All advertising materials mentioning features or use of this software
183      must display the following acknowledgement:
184
185         This product includes software developed by the Group for User
186         Interface Research at the University of California at Berkeley.
187
```

```
4. The name of the University may not be used to endorse or promote products
   derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ‘‘AS IS’’ AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
SUCH DAMAGE.
*/
```

B.2.3 MultiFileRecognizer.java

```
//// See bottom of source code for software license

import edu.berkeley.guir.lib.awt.geom.*;
import edu.berkeley.guir.lib.collection.*;
import edu.berkeley.guir.lib.gesture.*;
import edu.berkeley.guir.lib.satin.objects.*;
import edu.berkeley.guir.lib.satin.stroke.*;
import edu.berkeley.guir.lib.satin.event.*;
import edu.berkeley.guir.lib.satin.recognizer.*;
import java.io.*;
import java.text.ParseException;
import java.util.*;

/**
 * A wrapper for Rubine's Recognizer(enable multi files). This wrapper
 * looks at the absolute
 * bounding points of TimedStrokes, classifying on what the user sees as
 * opposed to the underlying stroke data.
 *
 * <PRE>
 * Revisions:  1.0.0  30-09-2002 Hideto Yamada
 *                    Created class MultiFileRecognizer
 * </PRE>
```

```
24    *
25    * @author <A HREF="http://www.cs.berkeley.edu/~jasonh/">Jason Hong</A> (
26    *         <A HREF="mailto:jasonh@cs.berkeley.edu">jasonh@cs.berkeley.edu</A> )
27    * @since   JDK 1.2
28    * @version Version 1.0.0, 06-16-1999
29    */
30   public class MultiFileRecognizer
31      implements SingleStrokeRecognizer {
32
33      //============================================================================
34      //===   NONLOCAL VARIABLES   ================================================
35
36      MultiFileClassifier classifier;       //// the Rubine's classifier
37      Vector classifiers;                ////  Classifier set
38      HashMap result;                      ////  Result set
39
40      //===   NONLOCAL VARIABLES   ================================================
41      //============================================================================
42
43
44
45      //============================================================================
46      //===   CONSTRUCTOR   =======================================================
47
48      /**
49       * Create a MultiFile Recognizer with an empty data set. Everything gets
50       * recognized as "tap".
51       */
52      public MultiFileRecognizer() {
53         classifiers = new Vector();
54         result = new HashMap();
55      } // of default constructor
56
57      //============================================================================
58
59      /**
60       * Create a MultiFile Recognizer.
61       *
62       * @param     rdr is a Reader reading the classification file.
63       * @exception IOException on a read error.
64       * @exception ParseException on a bad data file.
```

```
65       * @exception TrainingException on a problem with the classifier.
66       */
67      public MultiFileRecognizer(Reader rdr)
68         throws IOException, ParseException, TrainingException {
69         result = new HashMap();
70
71         //// 1. Create an empty GestureSet.
72         GestureSet gset = new GestureSet();
73
74         //// 2. Initialize the data files.
75         gset       = GestureSet.read(rdr);
76         classifier = new MultiFileClassifier(gset);
77         classifiers = new Vector();
78         classifiers.add(classifier);
79      } // of constructor
80
81      //===   CONSTRUCTOR   ============================================================
82      //=========================================================================
83
84      //===   ADD FILE  (Added by Yamada)=============================================
85      //=========================================================================
86      public void addFile(Reader rdr)
87         throws IOException, ParseException, TrainingException {
88
89         //// 1. Create an empty GestureSet.
90         GestureSet gset = new GestureSet();
91
92         //// 2. Initialize the data files.
93         gset       = GestureSet.read(rdr);
94         classifier = new MultiFileClassifier(gset);
95         classifiers.add(classifier);
96      }
97
98      //=========================================================================
99      //===   CLASSIFICATION   ==================================================
100
101     public Classification classify(TimedStroke stk) {
102        Classification c      = new Classification();
103
104        //// 0. Exception case.
105        if (classifier == null) {
```

```
106            c.put("tap", 1.0);
107            return (c);
108         }
109
110         //// 1. Get the timed polygon within the Timed Stroke.
111         ////    Get absolute in order to be scale-invariant. Otherwise,
112         ////    Rubine's fails, since it can't handle different sizes.
113         TimedPolygon2D  polyOld = stk.getPolygon2D(COORD_ABS);
114         int[]           xpts    = new int[polyOld.npoints];
115         int[]           ypts    = new int[polyOld.npoints];
116
117         for (int i = 0; i < polyOld.npoints; i++) {
118            xpts[i] = (int) polyOld.xpoints[i];
119            ypts[i] = (int) polyOld.ypoints[i];
120         }
121
122         TimedPolygon    polyNew = new TimedPolygon(xpts,
123                                                    ypts,
124                                                    polyOld.times,
125                                                    polyOld.npoints);
126
127         //// 2. Create a temporary Gesture to wrap around the TimedPolygon.
128         Gesture gesture = new Gesture();
129         gesture.setPoints(polyNew);
130
131         //// 3. Classify the Feature Vector.
132         SortedValueNumMap tmp_map, map;
133         try {
134            MultiFileClassifier tmpclassifier;
135
136             for (int i = 0; i < classifiers.size(); i++) {
137                 Set keys;
138                 Iterator it;
139                 Object key;
140
141                 tmpclassifier = (MultiFileClassifier)classifiers.elementAt(i);
142                 tmp_map = tmpclassifier.classify_satin(gesture);
143
144                 keys = tmp_map.keySet();
145                 it = keys.iterator();
146
```

```
147                 while (it.hasNext()) {
148                     key = it.next();
149                     result.put(key, tmp_map.get(key));
150                 }
151             }
152 
153             map = new SortedValueNumMap();
154             map.putAll(result);
155         }
156         catch (Exception e) {
157            map = new SortedValueNumMap();
158            map.put("Exception error", 1.0);
159         }
160 
161         //// 5.   Convert the result into standard Satin classification results.
162         //// 5.1. If there are no items, then return an empty classification.
163         if (map.size() <= 0) {
164            return (c);
165         }
166 
167         //// 5.2. Otherwise, scale the Mahalonbis distances by the max val.
168         double   maxVal = ((Number) map.getLargestValue()).doubleValue();
169         Set      keys   = map.keySet();
170         Iterator it     = keys.iterator();
171         Object   key;
172         double   val;
173 
174         while (it.hasNext()) {
175            key = it.next();
176 
177            //// 5.2.1. If we only have one value, then scale it to 100%
178            ////        no matter what.
179            if (keys.size() == 1) {
180               val = 1.0;
181            }
182            //// 5.2.2. Otherwise, scale it so that the largest value is 0%.
183            else {
184               val = ((Number) map.get(key)).doubleValue();
185               val = (maxVal - val) / maxVal;
186            }
187            // System.out.println("Key = " + key + ", Value = " + val);
```

```
188             // System.out.println("Keys = " + keys.toString());
189             c.put(key, val);
190         }
191 
192         return (c);
193 
194     } // of classify
195 
196     //===   CLASSIFICATION   ================================================
197     //=========================================================================
198 
199 } // of class
200 
201 //=============================================================================
202 
203 /*
204 Copyright (c) 2000 Regents of the University of California.
205 All rights reserved.
206 
207 Redistribution and use in source and binary forms, with or without
208 modification, are permitted provided that the following conditions
209 are met:
210 
211 1. Redistributions of source code must retain the above copyright
212    notice, this list of conditions and the following disclaimer.
213 
214 2. Redistributions in binary form must reproduce the above copyright
215    notice, this list of conditions and the following disclaimer in the
216    documentation and/or other materials provided with the distribution.
217 
218 3. All advertising materials mentioning features or use of this software
219    must display the following acknowledgement:
220 
221       This product includes software developed by the Group for User
222       Interface Research at the University of California at Berkeley.
223 
224 4. The name of the University may not be used to endorse or promote products
225    derived from this software without specific prior written permission.
226 
227 THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
228 ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
```

```
229  IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
230  ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
231  FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
232  DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
233  OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
234  HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
235  LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
236  OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
237  SUCH DAMAGE.
238  */
```