付録　文脈空間 API 案

　文脈空間は、主に自然言語処理で、会話的状況を解析し把握するために考えられた、データエリアである。イメージとフレームの両方を用いて、意味を把握するもので、Java の一つのクラス（ContextMap）で表現される。イメージは様々な情報を保持する３次元マップとか、論理のためならば n 次元マップとして計算機上に表現される。フレームは意味と意味のメトリックと値とからなる（意味とオブジェクトの場合もある）テーブルである。イメージを解析して記号としたものをフレームに持つので、イメージの方が沢山の情報をもっているといえる。だが、自然言語で使えるのはフレーム上の記号情報である。だから、必要に応じて、イメージを解析して、フレームに落として行くことになる。
　このようなイメージデータの扱いは自然言語に限らないはずである。パターン認識においても、一気に外界を解析して全ての情報を得ることなど、できないことである。なんらかの表象的な中間イメージにして、詳細は必要に応じて解析していくという形を取るだろう。文脈空間は将来重要な技術となると考える所以である。
　文脈空間のイメージから取り出したい情報は以下のようなものである。
・空間内のオブジェクト
・オブジェクトの属性
・オブジェクトの動作、操作、運動
・オブジェクトの動作、操作、運動の属性
・動作、操作、運動の格情報
・その他必要に応じて、充実させていく。

　これらの目標のために多くの API を用意する。API は、
・情報の設定
・情報の取り出し
・情報の操作（イメージ合成/分離、変形など）
・運動のシミュレーション
などを行う。

　空間内のオブジェクトが変形、移動した時には、そのオブジェクトに添付された情報もそのままオブジェクトに保持され、移動しなくてはならないことに注意したい。

１．概論
　文脈空間の操作のベースとなるクラスは ContextMap であり、その下に
・MapSystem・・・ボクセルイメージで各イメージを構成するピクセル毎に任意の型のデータが添付（connect）できる。
・PointerMapSystem・・・MapSystem 内の特定のピクセルセットを一つのマップシステムとして扱えるようにしたボクセルイメージ。MapSystem のサブクラスである。

・Frame・・・フレームのクラス
がある。これら４つのクラスが基盤クラスである。

文脈空間を操作する API は、目標に応じて次のような体系で設計する。
（１）メタ操作
文脈空間全体の動きを管理する API である。
・文脈空間の初期化、終了化
・文脈空間間のコネクションの創成、削除
・文脈空間のイメージを起動する。
・文脈空間のイメージを停止する。
・文脈空間のイメージの運動の同期を取る。

（２）オブジェクト創成管理
・イメージの創成、削除
・フレームの創成、削除
・アニメーションの創成、削除
・オブジェクトの属性操作

（３）オブジェクト運動操作
・回転運動
・直線運動
・運動軌道設定
・運動積分操作
運動のパスを積分する。
・運動差分操作
特定時間の運動による、イメージの変化を得る。
・運動属性（方向、速さ、加速度、力の大きさ）の操作

（４）オブジェクト変換操作
・射影変換
・アッフィン変換
・剪断
・拡大/縮小
・テンプレート変換
・変換パラメータの操作

（５）オブジェクトの運動と属性
これは、文脈空間の中核となる、パターン認識機構です。この処理のために文脈空間はあるのです。運動の属性、物体の属性、物体の配置や形などを解析します。
・STAY 操作群とその管理

・MOVE 操作群とその管理

（６）推論操作

これは、推論機能を実現 API 群です。プランニングとか、リーズニングで利用します。

・集合論演算

・論理演算

・パススルー（経路）操作

集合演算はマップシステムで、その上で、空き位置とか点が含まれる含まれないというようなことを実現します。

論理演算はフレーム間のコネクションに「～ならば～」などの意味を持たせて、検索型推論を実現します。

パススルーはタイル張り空間、これもマップシステムで実現して実現します。始点と目標点の間にタイルが敷き詰められているかを判断します。

（７）その他

・意味プリミティブごとにマップシステムを解析する API を設ける。解析や解析のための操作の対象は意味プリミティブ群である。

２．各論

API シグニチャの概要を以下に示していく。

２．１ メタ操作

２．１．１　文脈空間の初期化、終了化

ContextSpace cs=new ContextSpace(String id);　識別子 id の文脈空間を作る。

cs.close();　文脈空間を削除する。

２．１．２　文脈空間間のコネクションの創成、削除

Connection インターフェース； オブジェクトをコネクション子とする。

cs.connect(Connection c,Object o);

cs.delete(Connection c,Object o);

２．１．３　文脈空間のイメージを起動する。

ContextAttribute インターフェース；文脈空間属性を定義するインターフェース。

cs.start(ContextAttribute ca);

２．１．４　文脈空間のイメージを停止する。

cs.stop（ContectAttribute ca）;

２．１．５　文脈空間のイメージの運動の同期を取る。

cs.synchronizeWith(Context cs1,Context cs2,・・・);

２．２　オブジェクト創成管理

２．２．１　イメージの創成、削除

MapSystem　ms=cs.creatMapSystem(String　id,width,height,depth);識別子 id のマップシステムの創成
大きさは(width,height,depth)の３次元空間

PointerMapSystem　pms=cs.creatPointerMapSystem(String　id,Connection　c);識別子 id の部分マップシステムの創成（Connection　c を持つピクセルからマップシステムを作る）

a.close();マップシステムの削除

２．２．２　フレームの創成、削除

Frame インターフェース；フレームを定義するインターフェース。

Frame fm=cs.createTable(String id);　識別子 id のフレームを作る

b.close();

２．２．３　アニメーションの創成、削除

Animation インターフェースを持った MapSystem がアニメーション。

Animation a=ms.createAnimationMap(String id,width,height,depth);
識別子 id のアニメーションを定義する。マップシステム ms は親のキーフレームイメージになる。

a.add(MapSystem ms1);動画の一こまイメージを順次追加していく。

２．２．４　オブジェクトの属性設定

ms.connect(int x,int y,int z,Connection c,Object o); コネクトによるオブジェクトの設定
コネクトするのは座標(x,y,z)へ

ms.delete(int x,int y,int z,Connection c,Object o);

ms.setIntValue(int x,int y,int z, int value);整数を設定

ms.deleteIntValue(int x,int y,int z);

２．３　オブジェクト運動操作

２．３．１　回転運動

ms.rotate(int axis_x,int axis_y,int axis_z,double radian);軸の周りの回転を指定する。

２．３．２　直線運動

ms.move(int axis_x,int axis_y,int axis_z,double delta);軸方向に増分だけ移動する。

２．３．３　運動軌道設定

Trace t=new Trace(String id,int[] x,int[] y,int[] z,int sp,int listlength);Trace の識別子は id。線分列を指定して作る。

２．３．４　積分操作

MapSystem　targetMS=a.integral(int　start_image,int　end_image,Translation[] trans,double[] time_span　);　　各アニメーション要素イメージに変換（Translation）とその継続時間（time_span)を与えて,最終結果を得る。

２．３．５　運動差分操作

MapSystem delta=ms.getDelta(Translation trans,double time_span);

２．３．６　運動属性の操作

TransAttribute インターフェースで、属性フレームのひな形をつくる。

意味記号とメトリクスと設定値のテーブルである。

２．４　オブジェクト変換操作

２．４．１　射影変換

Translation インターフェースをもつ。

マトリックスの定義を行う。

２．４．２　アッフィン変換

Translation インターフェースをもつ。

マトリックスの定義を行う。

２．４．３　剪断

Translation インターフェースをもつ。

マトリックスの定義を行う。

２．４．４　拡大/縮小

Translation インターフェースをもつ。

ファクターの定義を行う。

２．４．５　テンプレート変換

Translation インタフェースをもつ。

テンプレート変換のメソッドは Process インターフェースを持つ。

Translation trans=new　TemplateTranslation(Process translationMethod);

２．４．６　変換パラメータの操作

Translation インターフェースで定義する。

マトリックスを定義することである。

２．５　オブジェクトの運動と属性

２．５．１　STAY 操作群とその管理

意味プリミティブ毎に定義する Animation セットである。

意味プリミティブとしては、次のものが考えられる。

・構造
・入れ子情報
・内外情報
・接続情報
・メトリック（角度、曲率、周波数）
・配置（connect,separate,beside,between,far）
・場所（東西南北、上下左右・・・）
・形態（akin,edge,close,open,cup,cap,flat）

・幾何（点、線、曲線、三角、四角、円・・・）
・動詞プリミティブ（help,use,stay,change,keep,get,give・・・）
・格情報

操作は、これらの設定（イメージの変形が起きる）と問い合わせをする API である。詳細は省略しています。

２．５．２　MOVE 操作群とその管理

意味プリミティブ毎に定義する Animation セットである。
意味プリミティブとしては、次のものが考えられる。
・パスの構造
・パスの入れ子情報
・パスの内外情報
・パスの接続情報
・パスのメトリック（角度、曲率、周波数）
・パスの配置（connect,separate,beside,between,far）
・パスの場所（東西南北、上下左右・・・）
・パスの形態（akin,edge,close,open,cup,cap,flat）
・パスの幾何（点、線、曲線、三角、四角、円・・・）
・動詞プリミティブ（move,keep）
・格情報

操作は、これらの設定（イメージの変形が起きる）と問い合わせをする API である。詳細は省略しています。

２．６　推論操作

２．６．１　集合論演算

MapSystem のサブクラスである。集合論理（集合論理プリミティブセット）API を持つ。
SetMap　set=new SetMap(String id);　要素数が不定の集合の場合
id はこのマップシステムの識別子
SetMap　set=new SetMap(String id,int elementNo,・・・);要素数が固定の n 次元マップで表現される集合の場合
set.addElement(Object o); 要素の追加
set.addElement(Object o,int px,int py);座標(px,py)に要素を追加
set.getElementNumber(); 要素数を得る。
set.getAllElement() ;全ての要素を得る。
set.getRestOf(SetMap oset); 指定の集合マップの要素を抜いた残りの要素を得る。

２．６．２　論理演算

MapSystem のサブクラスである。論理演算（「・・・とき」、「・・・ならば」など）API を持つ。

前件用のエントリに条件フレーム群を登録する。コネクションの種類で and、or、not を表現する。また条件の種類（「・・・ならば」とか）もコネクションの種類で示す。

後件用のエントリに結果フレーム群を登録する。コネクションの種類で、and、or、not を表現する。

LogicMap logic=new LogicMap(String id);識別子 id のロジックマップを創成する。

logic.addPreposition(Connection c,Frame preframe);前件の登録

logic.addPostposition(Connection c,Frame postframe);後件の登録

logic.isMatch(Frame　matchframe);パターン（フレーム）マッチング（true,false を返す）

２．６．３　パススルー（経路）操作

MapSystem のサブクラスである。集合演算と論理演算と追加 API を持つ。

PathMap path=new PathMap(String id,int width,int height,int depth);
識別子 id で大きさ（width,height,depth）の経路マップシステムを創成する。

path.setIntValue(int x,int y,int z, int value);升を数値で埋める。

path.connect(int x,int y,int z,Connection c,Object o);升にオブジェクトをコネクトする。

path.delete(int x,int y,int z,Connection c,Object o);コネクトを削除する。

path.hasPath(int sx,int sy,int sz,int ex,int ey,int ez,int value);始点から終点までにパスがあるかを問う。value と同じ値をもつ升目が始点から終点まで続いていたら、true。それ以外は、false。

２．７　その他

２．７．１　ピクセルを選択して新しいマップシステムを作る。

MapSystem newms=ms.extract(Connect c,Object o);同一の情報を持つピクセル群を新しいマップとする。

MapSystem newms=ms.extractByValue(int value);同一の情報（value）を持つピクセル群を新しいマップとする。

２．７．２　意味プリミティブの操作

ObjectPointer op=ms.getObjectPointer(Connect c,Object o);イメージ中のオブジェクトを指す情報を得る。

ms.attachTo(ObjectPointer op,int x,int y,int z);MapSystem 内の ObjectPointer で指されたオブジェクトを点(x,y,z)にあるオブジェクトに接触させる。このとき、イメージデータの変形を行う。

ms.detachFrom(ObjectPointer op,int x,int y,int z);MapSystem 内の ObjectPointer で指されたオブジェクトを点（x,y,z）にあるオブジェクトから離す。このとき、イメージデータの変形を行う。

２．７．３　文脈空間のマージ

ContextSpace newcs=cs.add(int x,int y,int z,Connection c,Context addcs);

既存の文脈空間の座標（x,y,z）にコネクション c を付してもう一つの文脈空間をマージさせて、新しい文脈空間を作る。

ContextSpace newcs=cs.delete(Connect c);

文脈空間から、コネクションｃでマージしている文脈空間を削除する。

２．７．４　内容検索

Frame searchframe=new Frame();検索項目を設定するフレームを創成する。

searchframe.addKItem(Object　keyobject1,Object　keyobject2,・・・);検索条件キーを指定する。引数は and 条件で、addKItem 毎に or 条件になる。

searchframe.addQItem(Object　keyobject);検索項目を指定する。これに対応しているデータを検索してくることを示す。

cs.search(Frame searchframe); フレームに設定されている意味項目の内、値が設定されていないものを、キーにして検索し、その項目の回答欄に設定する。

２．７．５　イメージの取り込み、取り出し

MapSystem ms=cs.createMapSystem(String id,Image image);

ContextSpace に指定して、識別子 id のマップシステムを作る。取り込むのは Java の Image オブジェクトである。

Image　image=ms.getImage(String　id);マップシステムから識別子 id のイメージを Java の Image 形式で取り出す。

３．パターン認識プロセス

パターン認識は基本的に意味プリミティブ毎にプロセスを起こし、学習して獲得する意味については、コマンドパターンを適用して、意味プリミティブを編み込んで実現していきたい。

機構として持つものは、次のものである。

・プロセス管理テーブルとプロセス群
・意味記号からプロセスを検索する、インバースインデックス
・ピリミテイブパターン群（各パターン認識プロセスによって定義する）
・高次の学習で得たパターン群（コマンドパターンで実現）

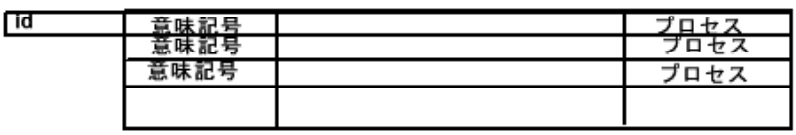


図３．１プロセス管理テーブル

意味記号	内容ポインタ	内容ポインタ	・・・
意味記号	内容ポインタ	内容ポインタ	・
意味記号	内容ポインタ	内容ポインタ	・・・

・・・・・・・

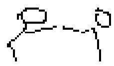
図３．２インバースインデックス

```
[歩く]

move
attribute:attach ground,swing hands, ・・・
attribute:speed (rapid than:ナメクジ、・・・)
               (slow than:car,・・・)

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・
case:(to:place)

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・
```

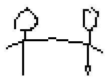
図３．３学習によって獲得するパターンの記述

例：「輪くぐり」の解析

（例文１）「二人の子どもが手をつないで向かい合った」



（例文２）「二人の人間が片手をつなげた」



例文１も例文２も、地面とを加えて考えると、輪ができることが内的パターン認識で記号化される。

知識により、「輪」は通り抜けられるものと評価され、その遊び化としての、「輪くぐり」が定義される。このように、どんな組み合わせでも、抽象的な意味プリミティブ「輪」が同定できれば、「輪くぐり」が解析により推論できるのである。

以上