これから、ＲＤＢベース人工知能の設計書を作っていく。その第一段階として知識ベースを詳細設計していく事とした。知識ベースは次のコンポーネント群からなる。

（１）知識管理
（２）学習システム
（３）意志・評価・プランニング
（４）数学的能力

Ａ．概論

人工知能が扱うデータは膨大であるため、また作業データも巨大であるため、永続化データ・・・データベースを基本に処理を実現していかねばならない。インコアで扱うデータは極力抑え、多くはデータベースへの直接処理でプロセスは実行されていくように設計する。

知識ベースの管理（知識管理）はデータベースの管理をしていくモジュールで、データの検索、データの永続化といった知識の基本アクセスを実現していく。永続化形式をインコア形式に変換するとか、その逆も行っていく。

学習システムは知識を体系化していく過程であり、記号ベースのカテゴリーシステムを実現し、知識体系を充実していく過程を担う。

意志・評価・プランニングシステムは知識を利用するという側面から管理する。人工知能として有利な知識はなにか、予定とか割り込み作業とかを管理し、人工知能を自律的に動かしていく基盤となるモジュールである。思考とか行動、言語行動、画像認識、画像生成といった作業をこの意志・評価・プランニングシステムにプラグインすることで、人工知能は完成されていく。外界に対して行動できるものになっていく。

数学的能力は、思考の基盤となるモジュールである。ただ、知識ベースと深く関連しているためここに組み込んでいく事とした。

Ｂ．モジュール設計

人工知能環境は AI モジュールによって構築するものとし、その配下の大きなサブモジュールはそのファクトリーメソッドで生成するものとする。

AI ai=new AI(String URL,String database_name,String user_name,String password);

URL,database_name,user_name,password:人工知能環境を保持する永続化エリアを指す。

List table_names=ai.getKnowledges();

List　table_names:現在在るテーブル群のテーブル名のリストを返す。

int date_time=ai.getCurrentTimeStamp();

現在のタイムスタンプを得る。

ai.terminate();　　人工知能を終了する。

１．知識管理

知識データベースへのデータの取り込みと知識データベースからデータを検索して取り出す処理を行う。

（１）永続化プロセス要素

知識データベースへのデータの追加を行う。

Knowledge　knowledge=ai.factoryKnowledge(String table_name);

String table_name:処理するデータベースのテーブル名

【説明】

知識データベースのアクセス準備をする。

knowledge.insertEntity(FrameData fd);

FrameData fd:ＤＯＭ形式のオブジェクトデータ

fd にはＸＭＬとなって保存されるデータの他にデータタイプやレコード識別子や重要度、タイムスタンプなどがデータとして付加されている。

【説明】

知識データベースにＸＭＬ形式でデータを保管する。

knowledge.insertLink(FrameData fd);

FrameData fd:ＤＯＭ形式のリンク（連想）データ

【説明】

知識データベースにリンク（連想）をＸＭＬ形式で保管する。

knowledge.replaceEntity(FrameData fd);

FrameData fd:ＤＯM形式のオブジェクトデータ
fd にはＸＭＬとなって保存されるデータの他にレコード識別子や重要度、タイムスタンプなどがデータとして付加されている。
【説明】
知識データベースにＸＭＬ形式でデータを保管(更新)する。

knowledge.replaceLink(FrameData fd);

FrameData fd:ＤＯM形式のリンク（連想）データ
【説明】
知識データベースにリンク（連想）をＸＭＬ形式で保管(更新)する。

knowledge.replaceEntity(FrameData fd,String field_name,String data);

FrameData fd:ＤＯM形式のオブジェクトデータ
fd にはＸＭＬとなって保存されるデータの他にレコード識別子や重要度、タイムスタンプなどがデータとして付加されている。
String field_name:更新するフィールド
String data:更新値
【説明】
知識データベースの所定のフィールドを更新する。

knowledge.replaceLink(FrameData fd,String field_name,String data);

FrameData fd:ＤＯM形式のリンク（連想）データ
String field_name:更新するフィールド
String data:更新値
【説明】
知識データベースのリンク（連想）の所定のフィールドを更新する。

FrameData fd=knowledge.getEntity(String id);

FrameData fd:取り出された知識データ
String id:取り出すレコードの識別子
【説明】
知識データベースから所定のレコードをＤＯM形式で取り出す。

FrameData fd=knowledge.getLink(String id);

FrameData fd:取り出されたリンク（連想）データ
String id:取り出すリンク（連想）レコードの識別子
【知識】
　知識データベースから所定のリンク（連想）レコードをＤＯＭ形式で取り出す。

（２）悉皆検索プロセス要素
パターンマッチングと連想による検索の２種類がある。
FrameDataSet fds=knowledge.getByPatternMatch(FrameData pattern_fd);
FrameDataSet　fds:検索結果をＤＯＭ形式のデータの集まりとして得る。
FrameData　pattern_fd:検索コマンド（パターンマッチング表記）を指定する。

FrameDataSet fds=knowledge.getByAsociation(String data_name);
FrameDataSet　fds:検索結果をＤＯＭ形式のデータの集まりとして得る。
String　data_name:連想の元となる知識データ要素の名前を指定する。

（３）悉皆走査プロセス要素
知識ベースのエントリーを順次検索していくことをサポートする。
ResultSet rs=knowledge.getAccessMethod(String field_name,String data);
ResultSet rs:リレーショナルデータベースの ResultSet を返す。
String field_name:検索のフィールド名
String data:検索するフィールド値

FrameData fd=knowledge.getNext(ResultSet rs);
FrameData　fd：ResultSet からレコードを順次取り出し、解析してＤＯＭ形式にして返す。レコードが無くなれば null を返す。

（４）発火

Knowledge fired_knowledge=knowledge.fire(FrameData fd,String[] table_names);

Knowledge fired_knowledge:発火した知識へのポインタ（リンク）を集めた知識セットを永続化（名前は fired「タイムスタンプ」）して返す。
また発火した各テーブルとレコードのタイムスタンプを更新する。
FrameData fd:発火の基点
String[] table_names:発火するテーブルを限定する場合に指定する。

２．コーパス管理

（１）コーパス管理プロセス要素の生成

Corpus corpus=ai.factoryCorpus();

コーパスを打ち立てる。

（２）コーパス知識ベースの登録

corpus.addKnowledge(Knowledge knowledge);

Knowledge knowledge:コーパスとなる知識ベース

KnowledgeSet ks=corpus.getKnowledge();

KnowledgeSet ks:コーパスとなる知識ベース群を得る。

３．モデル管理

（１）モデル管理プロセス要素の生成

Model model=ai.factoryModel();

モデルを打ち立てる。

（２）コーパス知識ベースの登録と取り出し

model.addKnowledge(Knowledge knowledge);

Knowledge knowledge:モデルとなる知識ベース

KnowledgeSet ks=model.getKnowledge();

KnowledgeSet ks:モデルとなる知識ベース群を得る。

４．カテゴリー管理

（１）カテゴリー管理プロセス要素の生成

Category category=ai.factoryCategory();

カテゴリーを打ち立てる。

（２）カテゴリー知識ベースの登録と取り出し

category.addKnowledge(Knowledge knowledge);

Knowledge knowledge:カテゴリーとなる知識ベース

KnowledgeSet ks=category.getKnowledge();

KnowledgeSet ks:カテゴリーとなる知識ベース群を得る。

５．コンテキスト管理

（１）コンテキスト管理プロセス要素の生成

Context context=ai.factoryContext();

コンテキストを打ち立てる。

（２）コンテキスト知識ベースの登録と取り出し

context.addKnowledge(Knowledge knowledge);

Knowledge knowledge:コンテキストとなる知識ベース

KnowledgeSet ks=context.getKnowledge();

KnowledgeSet　ks:コンテキストとなる知識ベース群を得る。

６．オントロジー管理

（１）オントロジー管理プロセス要素の生成

Ontrogy ontrogy=ai.factoryOntrogy();

オントロジーを打ち立てる。

（２）オントロジー知識ベースの登録と取り出し

ontrogy.addKnowledge(Knowledge knowledge);

Knowledge knowledge:オントロジーとなる知識ベース

KnowledgeSet ks=ontrogy.getKnowledge();

KnowledgeSet　ks:オントロジーとなる知識ベース群を得る。

７．学習システム

知識管理システムを使って、そこに解析データを保存し、データを分類していくのが学習システムである。

（１）共起関係パターン発見

タイムスタンプの同じ知識データを関係づけていく処理と共起関係にある知識要素をカテゴリーとして纏めていく処理とがある。この処理はデーモンとして人工知能が動いている間生きているものである。

タイムスタンプの同じデータは基本的に同じステージ、同じシーン、カットのコーパスとして関係づけられているはずである。この場合には、共起パターン発見処理は同じコーパス内の知識要素群がまだ他のコーパス内で共起しているかどうかを発見していく処理になる。つまり、次のようなテーブルを作っていく処理になる。

事象	共起事象（タイムスタンプ）	共起事象（タイムスタンプ）	共起事象（タイムスタンプ）	共起事象（タイムスタンプ）	・・・
事象 a	true	false	true	false	
事象 b	true	true	true	true	
・・・					

TimeStampAnalizer tsa=ai.factoryTimeStampAnalizer();

TimeStampAnalizer tsa:タイムスタンプの関連の解析（共起パターン発見）処理を打ち立てる。

Knowledge table=tsa.analize(FrameDataSet fds);

Knowledge table:タイムスタンプ解析結果のテーブルを永続化知識要素として得る。

FrameDataSet fds:共起関係を調べる対象の知識要素セットを指定する。

（２）and 要素を持った知識要素を得る。

FrameData fdo:基準知識要素

FrameData fdr:参照知識要素

FrameData fd=fdo.exclude(FrameData fdr);

FrameData fd:and 要素の知識要素を生成する

（３）or 要素を持った知識要素を得る。

FrameData fdo:基準知識要素

FrameData fdr:参照知識要素

FrameData fd=fdo.expand(FrameData fdr);

FrameData fd:or 要素の知識要素を生成する

（４）連想を設定する。

FrameData fdo:基準知識要素

FrameData relation:連想属性知識

FrameData fdr:連想する知識要素

fdo.associate(FrameData　relation,FrameData fdr);

連想の実施

（５）プロセス（コマンドパターン）の学習

データだけでなく、手続きも学習して充実していけなくてはならない。基本的に「思考システム」が学習データを提供していくことになる。思考の経路をトレースするようなコマンド履歴をコーパスとして記録していき、それをデータとして知識化していく過程がプロセスの学習である。そのため、データの学習と同じような機構を作り込めばよいことになる。ただ、思考システムとか行動システムがコマンドパターンで動作するようなプロセスオブジェクトを装備することが前提である。そのため、この機能は各プラグインのところで設計していくことになる。

（６）学習デーモン（最上位学習モジュール）

Learner learn=ai.factoryLearner();

learn.execute(Knowledge work_knowledge);

Knowledge work_knowledge: 作業中の知識データ群を整理して既存の知識体系に組み込んでいく

８．評価

（１）評価システムの設立

Evalator evalator=ai.factoryEvalator();

（２）評価の実行

int value=evalator(FrameData fd,Knowledge value_knowledge);

int value:評価値

FrameData　fd:評価対象の知識要素。これからの連想知識群によって評価を下す。

Knowledge　value_knowledge:評価基準を設定してある知識セット

【説明】

評価基準はオントロジーとしてひとかたまりの知識要素群になっている。

・重要度
・好き嫌い度
・緊急度
・向かうべき事柄、避けるべき事柄度

こんな評価が意味記号体系として存在していて、知識要素はある文脈のもとにこの評価データに連想関係を貼っている。文脈は連想関係を記述する知識要素の意味記号として設定されている。

これらデータをアクセスして、膨大な発火要素群から指定の事象の評価値を計算していく。

9．プランニング

（1）プラニングシステムの設立

```
Planner planner=ai.factoryPlanner();
```

（2）プランの実行

```
FramDataSet action_sequence=planner.execute（FrameData fd,・・・・・）;
```

FrameDataSet action_sequence:選択した行動を返す。

FrameData fd:重要度、行動の継続、緊急性、予定行動の情報を持った各行動シーケンスを FrameDataSet に保存した知識要素

【説明】

行動はどう展開していくべきか。それは一つのメインとなるシーケンスに誤差が大きいとどう行動を分岐していくかが記述された一連の知識としてあるのである。

評価システムからの結果を受けて、最適な行動を選んでいく。選ぶ行動は、意志システムに登録されている行動、思考知識要素が対象になる。

緊急処理は評価システムを経ずにプランニングが処置する。また誤差が大きくて選択する行動、思考は評価システムを経ずにプラニングで自動決定していく。

プランニングはまた、行動と思考の処理過程を監視している。

１０．数学的能力

数学的能力は集合演算、対応付け、カウンタである。集合は集合オントロジーがあって、そこに知識要素を連想づけることによって集合演算を実現できるようになっている。集合演算には順序付けとかカウンタがある。

数学的能力は「３本の矢」、「幾つ木が在るか？」などの文章から発火されて起動するものである。画像では、オブジェクトを見いだしたら本能的にカウントするだろう。そういうものが数学的能力である。基本的に知識ベースのプラグインが用意するプロセスによってこの数学的能力は起動される。

（１）照合演算の設立

SetProcess set_ontrogy=　ai.factorySetProcess();

（２）対応付け

set_ontrogy.setElement(String id,FrameData fd);

String id:登録する知識要素 FrameData fd に付ける識別子

（３）順序づけ

set_ontrogy.setSequnece(String id,int seqence_number);

String id:登録されている知識要素の識別子

int sequence_number:設定する順序番号

（４）カウント

int value=set_ontrogy.getSize();

int value:集合の要素の数

（５）パターンマッチング

String[] result=Matcher.extructStringsIn(String[] checkstrings,FrameData ofd);

FrameData ofd:文字列があるかどうかを探す対象の知識要素

String[] checkstrings:在るかどうかを検査する文字列セット

String[] result:知識要素に存在した文字を纏めて返す。

boolean match=Matcher.compare(FrameData rfd,FrameData ofd);

FrameData ofd:比較対象の知識要素

FrameData rfd:マッチするか検査するパターンを指定する。

boolean match:マッチすると true を返す。

【説明】

パターンのＸＭＬの表現は次の通りである。

知識要素群とパターンマッチングしていく入力データの表現は基本的にＸＭＬを採用していくとして、問題は相対アドレッシングと and,or,not 条件です。表現例として、

（例１）<ND><means>+human</means></ND><XX><surface>が</surface></XX>
などが考えられます。その他、
・並びは、{,}
・選択は、{|}
・否定は、！
・順序は<sequence>・・・</sequence>
・ルートは/で、相対は//
・間にテキストがあると、*

１１．プラグイン

人工知能に文章認識機構とか文章生成機構、行動、思考、パターン認識などの応用プロセスを追加し、プランニング機能から使えるようにする。追加機構は Module インターフェースを備えている物とする。

Module sentenceRecogniser=new SentenceRecogniser(AI ai);
モジュールオブジェクトを生成する。

sentenceRecogniser.execute(FrameData fd)
モジュールの実行は execute メソッドにデータエリアを指定して行う。ここにコマンドが設定されているものとする。

planner.setModule(String[] means,Module sentenceRecogniser)
プランニングシステムにモジュールを登録する。
String[] means:このモジュールを実行するきっかけとなる意味記号をしていする。

C．自律人工知能プログラム

人工知能システムは AI 以下の各クラスを使って、デーモン（エージェント）として作っていく事になる。

おわり