青葉設計書６　　2013/07/23

青葉思考システムの立式関連のデータエリアとプロセスについて記述します。結構複雑なので、備忘録としての意味もあります。

１．　基本データエリア

単語（変数）とか意味記号（オントロジー）とか格情報を保持するための基本データは Node で表現します。これは構造を持ったオブジェクトですが、対応するテキスト表現は XML です。ただ、タップルでなくて、括弧を使って構造を表現し、そのため、引用符は使っていません。<XXX p=1>YYY</XXX>は(XXX,p=1;YYY)と表現しています。この方がテキストを Node 形式にするのに処理が簡単で、高速になるからです。

ＲＤＢでデータを保存するときには、Node データはＲＤＢのレコードの一つのフィールドになります。そのほかに、プライオリティとか、タイムスタンプ、データ形式などの情報をもつレコードがＲＤＢの基本単位となっていて、それのオブジェクト構造形式のデータは FrameData です。FrameData は内部に Node を含みます。

このように、いくつかのオブジェクトの形式を統一して扱うために、それらを一つのオブジェクトとして処理するためのオブジェクト ObjectReferent というものを設けています。意味ネットワークはこの ObjectReferent を基本単位として構築しています。必要に応じて、Node を取り出して使っていきます。

２．　意味ネットワークデータエリア

日本語文章や図面を解析して、意味処理をした結果は意味ネットワークに落とします。意味ネットワークは次のオブジェクトの対で表現します。

（１） ObjectReferentNetNodeCluster とその集合 ObjectReferentNetNodeClusterSet

・点（point_ontology）と線(line_ontology)の連接で表現されるネットワークです。イメージの構造を表現します。

（２） ObjectReferentSentence とその集合 ObjectReferentSentenceSet

・単語（変数）の意味とか格関係情報を保存します。表現は Node の羅列です。

・単語を表現する Node を owner とし、その単語の持つ属性、格も Node で ObjectReferentSentence に保存する。

数式を扱うために、公式集も持ちます。公式集の変数と実際の問題が持つ変数に対応付けが必要になりますので、公式集も意味ネットワークで表現します。求める変数で公式集と意味ネットワークを検索し、関連する式を得て、その変数を問題文に対応させて、立式します。

３．　データ形式の例

（１）オントロジー記号の定義

意味ネットワークの属性定義でオントロジーの意味を定義します。

```
//
// ontology definitions (ontology data exist being coupled with formula data by an agent_case object
//
// attribute;------knowledge ontology----------------------------------------------------------------
//      length_ontology:(from_case;point_1)(to_case;point_2)
//      velocity_ontology:(change_case;length_ontlogy)(devider_case;time_ontology)
//      force_ontology;(change_case;velocity_ontology)(devider_case;time_ontology)
//
//      mass_ontology:
//      time_ontology:
//      space_ontology:
//
//      vector_ontology
//      pi_ontology
//      e_ontology
//      i_ontology
//
```

（２）式の定義

```
//
//   formula definitions
//
//   make a triangle formula;(setMeanName;triangle_ontology)
//                                   (makeFocusToStage;(object_case;triangle_ontology))
//                                   (makePointNodeAsOrigin;(object_case;triangle_ontology))
//     ・・・・//(macro_makeTriangle;
//         (makePointNodeAtArea;(object_case;pa)(at_case;triangle_ontology))
//         (makePointNodeAtArea;(object_case;pb)(at_case;triangle_ontology))
//         (makePointNodeAtArea;(object_case;pc)(at_case;triangle_ontology))
//          (makeLineNodeBetweenTwoPoint;(object_case;lab)(from_case;pa)(to_case;pb))
//          (makeLineNodeBetweenTwoPoint;(object_case;lbc)(from_case;pb)(to_case;pc))
//            (makeLineNodeBetweenTwoPoint;(object_case;lca)(from_case;pc)(to_case;pa))
```

//　　　　　　　　　　)
・pa,pb,pc は三角形の各頂点の名前です。
・lab,lbc,lca は頂点を結ぶ辺です。
//
//(macro_makeAttribute;
//　(setAttribute;(agent_case;pa)(have_case;angle_ontology)(angle_ontology_name;aa))
//　(setAttribute;(agent_case;pb)(have_case;angle_ontology)(angle_ontology_name;ab))
//　(setAttribute;(agent_case;pc)(have_case;angle_ontology)(angle_ontology_name;ac))
//　　　　)
・aa,ab,ac は各点 pa,pb,pc の持つ（have_case）角度です。

//
// make two velocities addition ontology;
//　　　　　　　　　　(setMeanName;velocity_ontology)
//　　　　　　　　　　(makeFocusToStage;(object_case;velocity_ontology))
//　　　　　　　　　　(makePointNodeAsOrigin;(object_case;ground))
// (macro_makeSean;(makePointNodeAtArea;(object_case;river)(at_case;ground))
//　　　　　　(makePointNodeAtArea;(object_case;boot)(at_case;river))
//　　　　)
・「陸」オブジェクトを作り、「陸」の上に「川」を、「川」の上に「ボート」を表わす点を作ります。

//(macro_makeAttribute;
//　　(setAttribute;(agent_case;river)(on_case;ground)
//　　(field_case;velocity_ontology)(velocity_ontology_name;v1))
//　　(setAttribute;(agent_case;boot)(on_case;river)
//　　(field_case;velocity_ontology)(velocity_ontology_name;v2))
//　　(setAttribute;(agent_case;boot)(on_case;ground)
//　　(field_case;velocity_ontology)(velocity_ontology_name;v))
//　　　)
・場として、「陸」に対して「川」の速度、「川」に対して「ボート」の速度、そうして、「陸」に対する「ボート」の速度を定義します。公式集の ObjectReferentSentence に設定する情報です。ここには、Ｖ＝Ｖ１＋Ｖ２という公式も設定します。

//

```
// attribute;------knowledge formula----------------------------------------------------------------
//      length:(_case;length_ontology)(length_ontology_name;l)
//      velocity:(_case;velocity_ontlogy)(velocity_ontology_name;v)
//      time:(_case;time_ontlogy)(time_ontology_name;t)
//---[l=v*t]
//      formula:(_case;formula_ontlogy)(formula_ontology_case;
//                          (=_case;(variable_case;l)(*_case;(variable_case;v)(variable_case;t)))
//---[f=v/t]
//      force;(_case;force_ontology)(force_ontology_name;f)
//      formula:(_case;formula_ontology)(formula_ontology_case;
//                          (=_case;(variable_case;f)(/_case;(variable_case;v)(variable_case;t)))
//

//--- [v=v1+v2]
//      formula:(_case;formula_ontology)(formula_ontology_case;
//              (=_case;(variable_case;v)(+_case;(varibale_case;v1)(variable_case;v2)))
//
//
//
```

４． 立式

問題は変数を求める文に集約します。運動を求めるのも、変数の時間的変化を求めることです。

（１） 問題文の変数は実変数です。これのオントロジーから、意味ネットワーク上の変数の所在と公式集上の仮変数の所在を得ます。

（２） 仮変数で、公式が定義されていますから、そこから式を得ます。変数を実変数に変換しないといけません。意味ネットワークと公式集の意味ネットワークをマッチングさせます。格情報を中心にして、変数対応を取ります。構造も意味ネットワークで表現されています。

（３） 実変数で式を得ます。隠れた変数にも実変数を割り振ります。

おわり