青葉設計書５　　2013/05/05（日）

短いプロセスを何度も高速に実行しなくてはならなく、並行処理が必須なのはパターンマッチング処理ですね。特に、人工知能はこのパターンマッチングが何事においても頻発します。

（１） 図面認識ですと、線画解析結果から意味オブジェクトネットワークを作るとき、パタンマッチングを使います。

（２） 日本語認識ですと、形態素解析から格情報を明確化した意味表記（Prolog 表記）にするところ、また意味オブジェクトネットワークを作るときに、パターンマッチングを使います。

（３） 思考ではプロダクションシステムの実行にパターンマッチングを使います。

（４） 事象の解析・認識（図面認識とか日本語認識）の途中で、色々な知識を連想して発火します。また細かなプロセスを沢山発火して解析していき、知識を得て行きます。これらの過程もパターンマッチングして行っていきますが、更に、連想や解析で得た結果の整合性を取るため、また、後の利用に使うため、データを管理していく必要があります。その整理作業にパターンマッチングを使います。

（５） その他にも、各プロセスでパターンマッチングは行われるでしょう。

なんか、データベースの上にパターンマッチング機構を持ったシステムはまとめてあると良いように思えます。それが「知識ベース」なのかもしれません。個人的には、パターンマッチングを抱える、画像処理部と自然言語処理部と思考部プロダクションシステムと思考部連想システムのそれぞれにＰＣを割り当てて、信濃プロジェクトを実現して行こうかと考え始めています。LIFEBOOK を４台ですね・・・調達していこうかと夢を抱いています。来年か・・・。再来年か・・・。

パターンマッチングの高速化案を「青葉設計書４」に書きましたが、更にもっと高速な方法を考えました。基本的には同じ考えですが、もっとグローバルに HashMap を使っていくのですね。つまり、

検索キーは普通、複数あります。これから、HashMap のキーを自動生成していくのです。それもソート済みのキー群を生成していく。そのキー群を１つのキーとして纏めて、MashMap のキーにしていく。バリューは実際の構造化したパターンのオブジェクトなわけです。取り出しにも、同じようなキー生成して HashMap 検索するのです。

これなら、マッチングパターン候補を得るのに、キーとなる記号から HashMap を検索するキーを自動生成するプロセスと、ハッシュによるアドレス検索の２段階だけで行えます。

もちろん、メモリ食うわけで、だから LIFEBOOK４台なのです。　　おわり。