

# Coron でない? み!

## 最終回 Coronと電磁石と 群ロボット(その2)の巻

春雨 忍次 イラスト:高木信孝

Coron ボードの中の妖精・コロンちゃんです。春になるとむくむくおきてくるのが「ロボット自作するぞ!魂」。

Coron にも拡張基板がそろってきました!株式会社ピルクスからは、Bluetooth 無線モジュールの「PBT-002」、テクノロジーからは Coron シスターズシリーズ「ジュンコちゃん」「アイちゃん」「サクラちゃん」「ナナちゃん」。「ジュンコちゃん」「アイちゃん」は LEGO MAINDSTORMS と Coron をつなぐ拡張基板、「サクラちゃん」「ナナちゃん」は待望のシリアルサーボとの接続拡張基板です。

シリアルサーボなら、PWM の出力が 16 本しかない制限から解放されますね!

みなさんからの「こんなのつくってみたい」という連絡お待ちしています。

### 「自律ロボットプログラム=アルゴリズム」

前回、アルゴリズムについて説明しましたが、実際のプログラムに書き込んでいきましょう。

プログラムのスパゲティー化を避ける方法を紹介します。Java で書いても、Ruby で書いても書き方が悪ければいくらでも読みづらく、バグがやすいプログラムになってしまいます。C 言語はポインタによるメモリ直接上書きが可能です。

よく、プログラム=データ構造+アルゴリズムといわれますが、自律ロボットのプログラミングには、データ構造はほぼ必要ないので、アルゴリズムをそのままプログラム化すれば OK です。

まず、各状態があり、その状態同士を結びつける状態遷移図を書き表し、それをプログラムの形にする雛形をつくります。状態が増えたらその部分を足すだけで、拡張も楽に行えます。状態部分を書き換え、他のアルゴリズムへの変更も簡単かも(?)

### 「状態遷移図を書いてみる」

まず、ロボットの状態を書き出してみましょう。前回説明した、「待機」「検索」「合体」に加え、「エラー」という状態を追加しました。検索が見つからなかった時や、合体に失敗してしまった時にエラー状態になり、復帰を試みるようにします。

各状態から遷移する先に矢線(→)を書きます。双方向に遷移しても OK です。

各々の矢線には発生する条件があります。格好よくいうとイベント/アクションです。それを矢線に書き加えれば、より明確な状態遷移図になります。

このように状態遷移図を書くと、処理すべき状態や、状態が遷移する条件が見え、抜けもなくなります。

もちろん、論文のページ稼ぎにもなりません(笑)。

状態や条件に抜けがあるとロボットが無限ループに陥ったり、暴走することがありますが、状態遷移図をかけば未然に防ぐことができます。

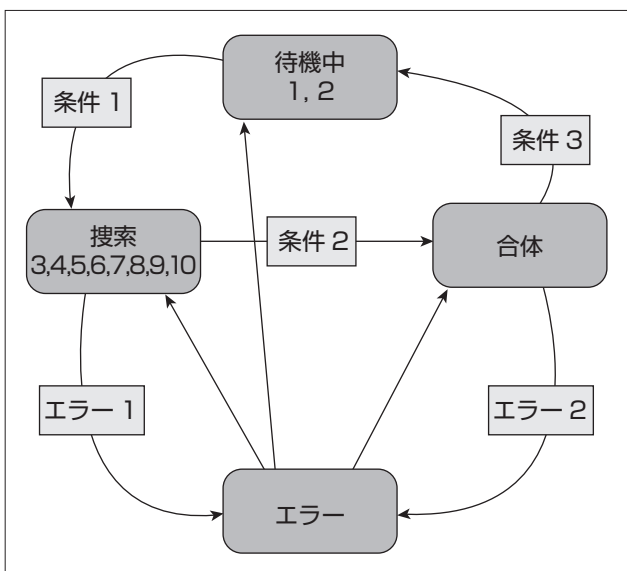
でも、人工知能プログラムは、状態遷移を自分で増やしていくことになるんでしょうねえ。

状態遷移図を書くときのポイントは、

状態(ステータス)は常に【○○待ち】か【○○中】

になることです。待機中は【音声入力待ち】、検索は【搜索中】、合体は【合体中】になるわけです。

前回のおさらいをしながら、状態遷移図の各状態内部処理について確認しましょう。L は、電磁石をつけているロボットの「コロレッジャー」、T はホールセンサをつけている「コロトラング」でしたね。



状態遷移図

### リスト: プログラムのメインリスト

```
void main(void)
{
    int status;
    while(1){
        switch (status){
            case 1:
                wait(); // 待機中
                break;
            case 2:
                search1(); // 搜索
                break;
            case 3:
                combine(); // 合体
                break;
            case 4:
                error(); // エラー
                break;
            default:
                break;
        }
    }
}
```