....

＊＊線画解釈プログラムの使用法と実験について＊＊

プログラムはJavascript で実装されており、 適当なWebブラウザでai.html を開くと実行できる。

主な操作は以下のとおりである。

頂点の追加： 画面や辺をクリックすることで頂点の追加ができる.

辺にカーソルが当っているときは辺が赤くなるためクリックの際の参考にされたい.

追加された直後の頂点はアクティブになっている.

辺の追加： 頂点をクリックすることでアクティブな頂点からクリックした頂点へ辺が引かれる.

また画面をクリッ クして頂点を追加した場合は自動的にアクティブな頂点から辺が引かれる.

頂点のデアクティベート： アクティブな頂点をクリックするとデアクティベートできる. と実装さ

以下のような機能も追加されている。

・Save、Load 機能（入力：[s][l]）

・Undo 機能（入力：辺を伴う描画を Undo[u]、ノードのみの描画を Undo[b]）

・グリッド描写機能（入力：表示[g]、非表示[r]）

・エスケープ機能（入力：[Esc]）

これらについてその仕様と実装方法を説明する。これらの入力は関数 key_event によって定義されている。一つ目は Save,Load 機能である。任意の入力後にキー「ｓ」を入力するとその状態を記憶する（セーブ)。その後「ｌ」を押すと、セーブ後に図形を描写したかどうかにかかわらずグリッド線も含めてセーブしたときの状態に戻す（ロード)。ロード直後にアクティブになっているのは、セーブ前に最後にプロットしたノードである。そのため、ロード後はセーブ時の続きから描画できる。ただし、セーブデータは一つのウィンドウ間でのみ保持され、リロードしたり別ウィンドウで開いた場合はセーブデータを読み込むことはできない。これは、キー入力「ｓ」があったときに、その時のノードと辺のデータ配列、グリッドが描画されているかをコピーし、「ｌ」があったときにそれまでの描画を一度破棄し、コピーしたデータをもとに一から描画している。描画をし直す関数は update_view_undo()、グリッドを描画する関数は write_grid()を用いている。それぞれについては後述する。二つ目は Undo 機能である。これは、「Undo したい操作が辺＋ノードの描写かノードのみの描写か」によって二種類のキー入力を設けた。前者はキーU、後者はキーB である。辺＋ノードの描写の場合は、それが多面体の最後の一辺であったならば辺のみを取り除き、描画途中であったならば辺とノードを取り除くようにした。この場合分けは、終点となったノードがすでに描画されていたものかどうかに依っており、描画されていたならばそのノードはラベル付けされるはずなのでその有無で判断している。取り除き方は、消したいものを格納した配列(edges/nodes)のうち、最も末尾のものを削除し、削除したデータをもとに再描画している。この再描画はロードと同様のことをしている。関数 update_view_undo は、保存されたノード、辺の配列要素について、それぞれの情報を読み込み描画するものである。三つめはグリッド描画関数である。キーG を入力するとグリッドが描画され、グリッド表示中はクリック場所に最も近いグリッドの格子点にのみプロットされるモードとなる。キーR を押すとグリッドは非表示となり、任意の場所に入力できるようになる。まず、ウィンドウサイズを取得し、変数 size で指定した幅の等差配列を作成する。その配列に基づいて罫線を描写する。その関数が write_grid である。表示

中であることを示すフラグを用意し、そのフラグに応じてノードの決定方法を制御した。表示中は等差配列の値をもとに最も近い格子点を求め、その点をノードとして入力することにした。最後にエスケープ関数であるが、ESC キーを押すとアクティブになっているノードをディアクティベートすることができる。これによって、再度ノードをクリックしなくて済む。これは、selected_node を未定義に変更し、画面を再表示することで実装されている。サンプル画面は Figure 1 である。

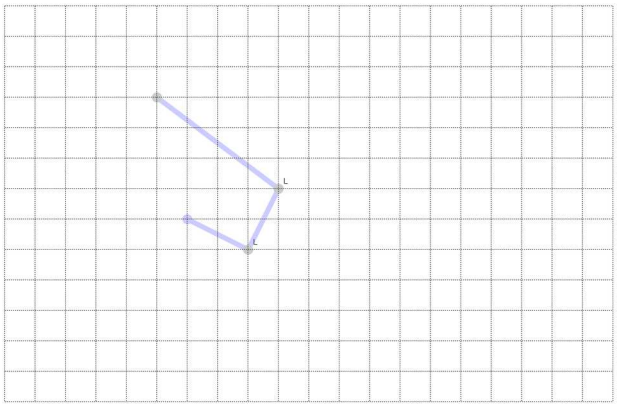

Figure 1　サンプル画面

このプログラムをもちいて不可能図形を描写し、その応答を見た。まず、Figure 2 のような描画をした。これは不可能図形であるが、図中の同色の印をつけた辺が矛盾していることからそのことがわかる。しかし、そのうち 2 色ずつは同一辺上にあるので同じ印でなければならない。各ノードは制約条件を満たしているので、プログラム的には間違った点はないが、各ラベルに私たちがつけた意味を考慮した時に矛盾が生じる。

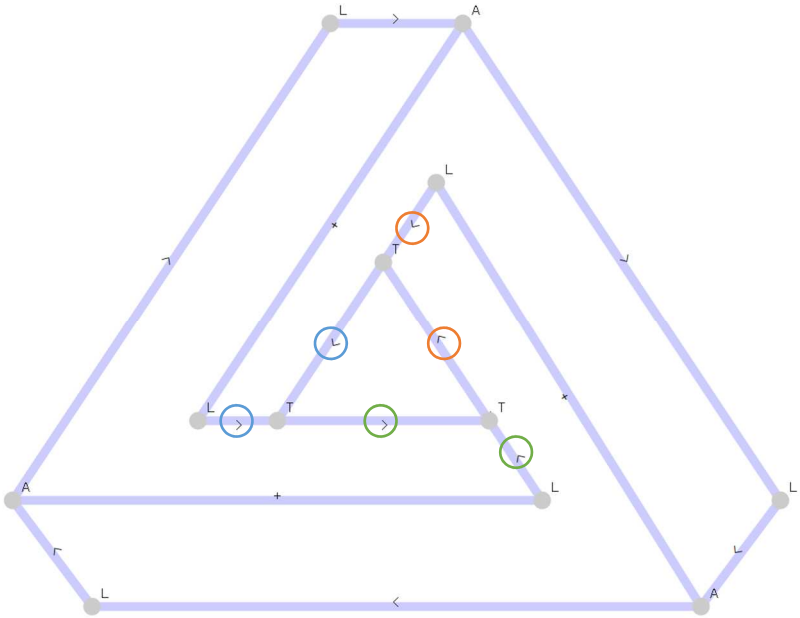

Figure 2　不可能図形の実験

また、Figure３のような不可能図形も描画した。これは、各側面の境界線の交点が一か所で交わらないため不可能図形であるが、プログラム上では実現可能な三角錐の一部と同様にラベル付けされている。これも、制約条件を満たしており、実際に実現可能な図形かどうか判断のためにはさらなる条件が必要であることがわかる。

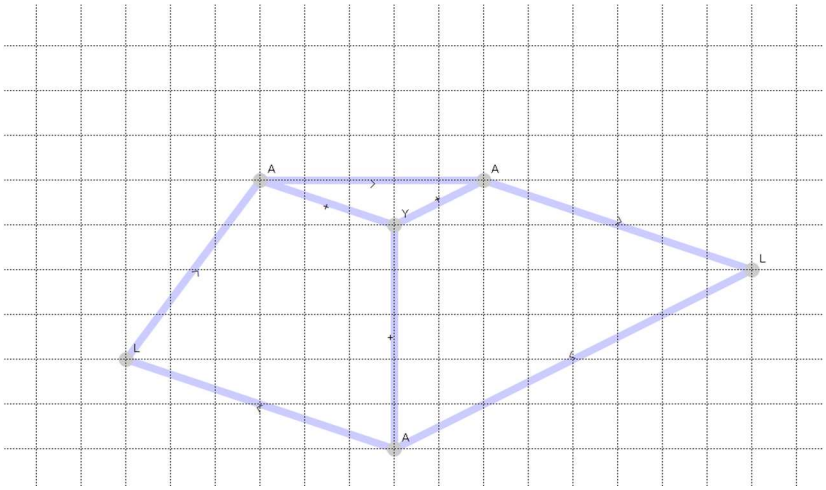

Figure 3　不可能図形の実験２

最後に、無限階段を実験した。簡単のため段数は少なくしたが、結果を Figure４に示す。この場合は、ノードは正しく表現されていたが辺がラベリングされなかった。よって、制約条件を満たすようなラベル付けができなかったと考えられる。

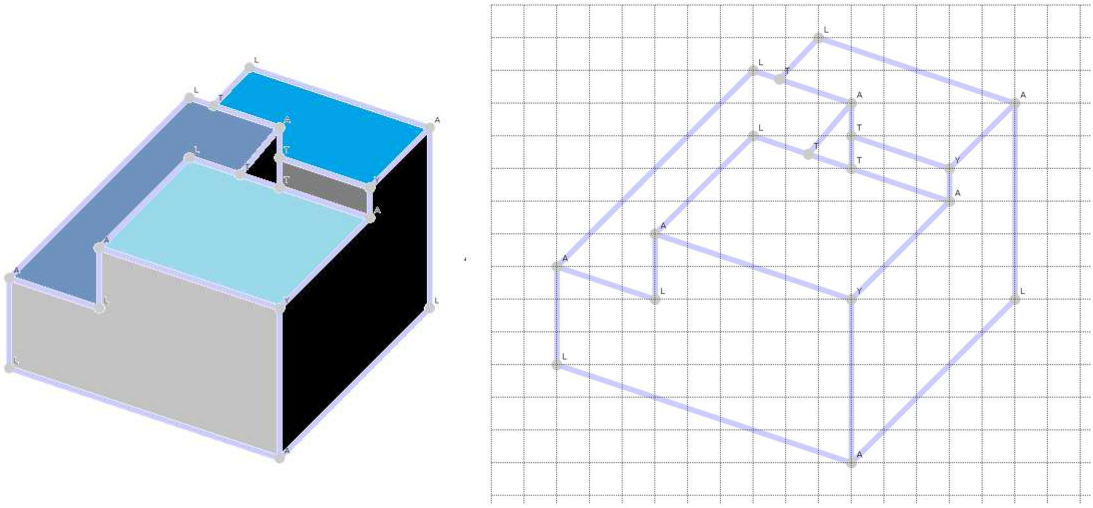

Figure 4　無限階段の実験

このように、不可能図形には二種類あると考えられる。Figure２，３のように制約条件を満たすがその意味付けを考えたときに矛盾するものと、Figure４のようにそもそも制約条件を満たしえないものとだ。後者に対しては人工知能（プログラム）は人間と同じ結論を導くが、前者に対してはそれ以外の条件を必要とする。

参考文献

（１）レポート課題：「制約伝播：線画のラベリング」資料
http://www.slis.tsukuba.ac.jp/~hiraga.yuzuru.gf/AI/label.shtml
（２）北岡明佳の錯視のページ「不可能図形」、北岡明佳、立命館大学 2002.5.10 開設
http://www.psy.ritsumei.ac.jp/~akitaoka/fukano.html
（３）「svg 要素の基本的な使い方まとめ」
http://defghi1977.html.xdomain.jp/tech/svgMemo/svgMemo_20.htm