知識工学と人工知能　48－6
（１９８６・９・１８）

解答の正誤を判定する回路知識の表現法

渡辺　成良　　兵藤　光夫　　福田　正弘

群馬大学　工学部

電気回路演習の解答文に含まれる誤りを発見するために必要な回路知識の表現法について述べる。与えられた演習問題に対して学生が示す解答文には、回路素子の接続関係と回路法則から導かれる線形方程式を解くために、適用される規則や法則の説明文と共に式の変形や方程式の解法などが含まれており、正解に至る手順が示されている。このような解答文の手順を追い含まれた誤りを発見して指摘するには、オームの法則、キルヒホフの電流及び電圧則、閉路方程式や節点方程式の立て方の知識に加えて、式の変形や連立方程式の解き方についての知識を必要とする。本文ではこのような知識の表現法とその使用法について述べ、その問題点を示す。

Knowledge Representation for Inspecting Solutions of Network Problems

Shigeyoshi WATANABE, Mitsuo HYODOU and Masahiro FUKUDA

Faculty of Engineering, Gumma University

Knowledge representation for inspecting solutions of network problems is shown in this paper. When a network problem is given to a student, he writes a solution process according to his knowledge of network analysis and algebra. To find mistakes in his solution process, knowledge about transformation of equations and solution method of algebraic equations is required in addition to knowledge of ohm's law, KVL, KCL, loop analysis and nodal analysis. In this paper, knowledge representation by Hyperware-Prolog which is a dialect of Prolog and is available for MC68000 processors under CP/M-68K is shown and usage of the knowledge and problems appeared in the execution are discussed.

1．　はじめに

電気工学のＩＣＡＩ（Intelligent Computer Assisted Instruction）システムとしては、ＳＯＰＨＩＥ[1]とＥＬ[2]が知られている。ＳＯＰＨＩＥは問題として提出された電子回路の故障箇所を学生が発見して直すのを手助けするシステムであり、相互に英語で会話し合いながら作業を進める。システムはその過程でみつかった学生の論理の矛盾や冗長さなどを指摘することができる。ＥＬは電子回路の問題解決システムを目指したもので、オームの法則やキルヒホフの法則に加えて回路のトポロジーや回路素子データを下に、前向き推論を実行して問題を解くことができる。

さて電気回路の学習は電気工学の基礎知識を得るために重要であり、学生は演習問題を解くことによって理解度や演算能力を増すように努めている。演習では学習によって得られた知識を上手に適用し、計算間違いをしないで出来るだけ短い時間で解を得る手順を学ぶことにあり、解へ至る手順を問題とせずに単に正しい解が得られればよいという考えでは、演習の効果は上がらない。従って一般には教官が答案を添削する方法が取られているが、これでは多数の学生を相手にすることが困難となるために答案添削を目的としたＩＣＡＩシステムの開発が望まれる。そのようなシステムは、

1）　解答手順に含まれる文や式を一つずつ追い、それぞれの意味を解釈して誤りを発見する
2）　手順の論理的矛盾や冗長さを発見し指摘する
3）　手順の行きずまりに対して適切な助言を行なうことが出来る
4）　理解度や演算能力を推定し、適切な演習問題を与えることが出来る

機能を有する必要がある。ここで、2）、3）、4）の機能は高度の推論機構を必要とする。また3）、4）の機能があれば、解答の途中でも助言が行なえるようになる。

本研究では1）の機能を有するシステムを試作したので[3],[4]、そのシステムで利用される回路知識と数式処理の表現法を示し、問題点を議論する。

2．　答案診断システムの概要

図1の回路でコイル l に流れる電流を求めた解答例を図2に示す。各素子の変数名と節点番号（素子と素子を接続するノードの番号）は回路図の中に図示されているものとし、文はローマ字分かち書きの表現とする。電圧や電流の変数名とそれらの向き、閉路（ループ）を構成する素子の集合とその向き等を解答者が定義する場合には、普通は回路図の中の適切な場所に変数名や矢印を書き加えるが、本システムでは文章で定義するものとした。また式の導出や変形にあたっては、どのような規則を用いたのかを説明する文がその式の前に書かれていなければならない。

このような解答文は図3の概念図で示されたシステムのファイルに入力される。日本語構文解析部は入力された文を一文ずつ意味解析し、定義、適用、結果の3つの格フレームのいずれかに分類する。格フレームの構造は、

```
[define, [[object, filler], [goal, filler]]]
[apply, [[object, filler], [instrument, filler]]]
[obtain, [[agent, filler], [result, filler]]]
```

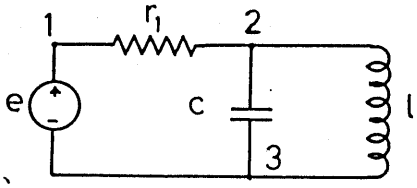

図1。　演習問題例
コイル l に流れる電流を求めよ。
電源 e の角周波数は w とせよ。

(1)　r1 wo nagareru denryuu wo i1 to suru.

(2)　i1 ha node1 kara node2 he nagareru to suru.

(3)　c wo node2 kara node3 he nagareru denryuu wo i2 to suru.

(4)　i3 ha l wo node2 kara node3 he nagareru denryuu to suru.

(5)　[r1, c, e] wo loop1 to suru.

(6)　loop1 ni kvl wo mochiiru to ex1 to naru.

(7)　ex1　e=r1*i1+i2/(j*w*c).

(8)　[l, c] wo loop2 to katei suru.

(9)　loop2 ni kvl wo tekiyousuru to　ex2　j*w*l*i3-1/(j*w*c)*i2=0　to naru.

(10)　node2 ni kcl wo mochiiru to ex3 to naru.

(11)　ex3　i1=i2+i3.

(12)　ex3 wo ex1 ni dainyuusuru.

(13)　ex4　e=r1*(i2+i3)+i2/(j*w*c).

(14)　ex2 wo i2 ni tsuite toku to　ex5　i2=j*w*l*i3*(j*w*c)　to naru.

(15)　ex5 wo ex4 ni dainyuusuru to
ex6　e=r1*(j^2*w^2*c*l*i3+i3)+j^2*w^2*c*l*i3/(j*w*c)　to naru.

(16)　ex6 wo i3 ni tsuite toku to　ex　i3=e/(r1*j^2*w^2*c*l+r1+j*w*l)　to naru.

図２。　図１の答案例

となっており、例えば図２の文（１）、（６）は

[[define, [[object, [i, [r1, dummy, dummy]]], [goal, i1]]]
[[apply, [[object, loop1], [instrument, kvl]], [obtain, [[agent, dummy], [result, ex1]]]]

と表現される。文（６）の例では解析結果は２つの格フレームのリストとなっている。

構文解析部で意味解析が完了した答案は統括部に渡され、定義と適用の格フレームは回路方程式導出部へ、また結果の格フレームは数式処理部へ送られて、それぞれの格フレームが正しく作られているかまたは格フレーム間の関係が正しいかが診断される。付録に示した例は、統括部が図２の答案例について診断していく過程を示したものであり、法則の適用によって方程式を導出したり、それを答案の式と比較したりする過程がよくわかる。

このように解答手順を追いながら、変数の定義や法則の適用が正しく行なわれているか、解答者が導出した式が

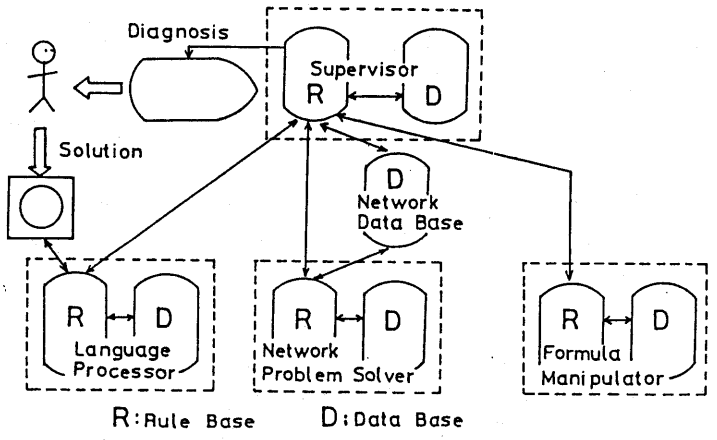

図３。　演習システムの概念図

間違っていないかを診断するシステムを作成するためには、回路方程式導出部や数式処理部の機能が十分でなければならない。

3．　回路知識の表現

演習問題の答が正しいかどうかだけを問題にする場合には、ＥＬのような問題解析システムで解いた答を解答者の答と比較するだけでよいが、解答手順の途中で生じた間違いを発見するためには、その手順に忠実に従いながら方程式を導出したり解いたりする必要がある。このため回路方程式導出部は、演習問題に関する回路情報を保持する回路情報データベース(Network data Base)と方程式を導出するための法則や規則を保持するルールベース及び方程式を導出する際に必要な回路データを保持するローカルデータベースから構成されている。

3．1　回路情報データベース

演習問題毎に作成され、素子毎の情報を定めるelement、角周波数を定めるfreq、素子に生じる電圧や流れる電流、合成インピーダンス、節点電圧をある値に設定する場合に用いられるsettingからなる。図４に例を示す。

```
element(e, e, voltage-source, 1, 3)
freq(w)
setting(i, current, 1, 2, 3, goal)
setting(p, 3, _, 0, _, nogoal)
```

図４。　回路情報データベースの例

3．2　ルールベース

ルールベースには

1）　初期化ルール
2）　定義診断ルール
3）　適用診断ルール

がある。初期化ルールは述語redefineにより、回路情報データベースの内容を初期化して図５のような述語を作成し、ローカルデータベースに保存する。ここでloop-counterは独立なループ数が２つあることを示している。

```
device(e, [e,0], voltage-source, 1, 3, [0,0], [e,0])
device(1, 1, inductor, 2, 3, [0, w*1])
nodes([3, 2, 1])
devicelist([r1, c, 1, e])
loop-counter(2)
node-counter(2)
voltage(_, e, 1, 3)
```

図５。　初期化ルールによる結果の例

さて統括部から送られる定義の格フレームは、図６に示された６つの述語のいずれかによって診断される。例えば、

```
check-loop(loop1, [r1, c, e], comment)
```

において、素子ｒ１，ｃ，ｅがループを形成し、この定義以前に定義されたループのいずれとも線形独立ならば、commentは'ok'となり、そうでない場合は適切な言葉が返される。'ok'の場合はローカルデータベースに

```
looplist(loop1, [node1| r1, c, e])
```

が登録される。同時にloop-counterの内容を１つ減少させ、正しく定義されたループの数をかぞえる。現在は完成していないが、統括部がこのローカルデータベースを調べることにより、解答者がどんな解析手法を用いているか、解析するために必要な条件を十分に設定できたかなどを知ることができ、１節で述べた２）や３）に対する推論機能を作成できると考えられる。

○ 閉路の定義

check_loop（ループ番号，素子リスト，Comment）.

looplist（ループ番号，［ノード番号(始点)｜素子リスト］）.

○ 枝電流の定義

def_cur（枝電流ラベル，素子ラベル，ノード１，ノード２，Comment）.

current（素子ラベル，枝電流ラベル，ノード１，ノード２）.

○ 枝電圧の定義

def_vol（電圧ラベル，ノード１，ノード２，Comment）.

voltage（_，電圧ラベル，ノード１，ノード２）.

○ 閉路電流の定義

def_loopcur（ループ番号，ループ電流ラベル，ループ内の1素子，ノード1，ノード2，Comment ）.

loopcurrent（ループ番号，ループ電流ラベル，ループ内の1素子，ノード1，ノード2 ）.

d_looplist(ループ番号，［ノード番号(始点)｜[素子1, N_{11}, N_{12}], ･･･, [素子n, N_{n1}, N_{n2}]] ）.

○ 節点電位の定義

def_pot（ノード番号，電位ラベル，Comment）.

potential（ノード番号，電位ラベル）.

○ 合成インピーダンスの定義

def_imp（合成インピーダンスラベル，接続構造，数式，Comment）.

impedance(合成インピーダンスラベル，素子リスト，ノード1，ノード2，数式）.

＊ 接続構造の例

- r1+r2//r3', 'r//c//l', 'r1//c1+(r2+l2)//c2'

図６。 定義診断ルール

適用診断ルールは、ｋｃｌ，ｋｖｌ，ｏｈｍの３つの式導出ルールを用いて適用の格フレームを調べ、正しい式を導出する。図７にルールの内容を示した。述語kclから得られるノード番号から以下のようにしてkcl式を導出する。

１） 節点番号の存在を確認する

２） その節点に接続されている素子を全て見付け出す

３） 見付け出された素子の枝電流を求め、その総和をとる。

枝電流を求める方法の検索手順は、枝電圧、枝電流、電流源の順序とし、枝電圧が定義されていればその素子のインピーダンス又は合成インピーダンスで割って求める。この順序で探索すれば、kclが節点電位法による解法に多く用いられるために、縦型探索の効率が良くなる。得られたkcl式は左辺＝０の式で表現されているものとして、左辺だけを作る。

kcl(ノード番号、 kcl式、 comment)

kvl(ループ番号、 kvl式、 comment)

ohm(素子ラベル、 ohm式、 comment)

図７。 適用診断ルール

述語kvlによって得られるループ番号から以下のようにしてkvl式を導出する。

1）　loop-listを検索して、指定されたループが存在することを確認する
2）　対応するloop-listから素子リストを取り出す
3）　各素子の枝電圧を求め、それらの総和をとる。
枝電圧を求める方法の検索手順は、枝電流、枝電圧、電圧源の順序とし、枝電流が定義されていればその素子のインピーダンス又は合成インピーダンスを掛けて求める。

述語ohmによって得られる素子ラベルから、
1）　素子の存在を確認し、枝電流とその基準方向を求める
2）　素子のインピーダンスを求め、枝電圧とその基準方向を求める
3）　枝電流と枝電圧の向きの関係を調べて、ｖ－ｉＺ＝０又はｖ＋ｉＺ＝０の左辺をohm式とする。

以上のような法則の適用診断ルールとloop-counter又はnode-counterを用いて閉路方程式や節点方程式を導出する。例えば、閉路方程式が正しく導かれているか、問題を解くのに十分な数の方程式が作られているかは、kvl式とloop-counterを調べることによって検証できる。

4．　数式処理部

数式処理部は解答者が導出した式と回路方程式導出部から得られた式を比較したり、解答者が行なった式の変形と元の式を比較する機能を有する。数式は以下のように一義的な構造に標準化される。

(係数＊文字列＾指数＊・・＊文字列＾指数＋・・＋係数＊文字列＾指数＊・・＊文字列＾指数)
／
(係数＊文字列＾指数＊・・＊文字列＾指数＋・・＋係数＊文字列＾指数＊・・＊文字列＾指数)

標準化された式は、分子の第一項が正の実数となるようにしてあり、かつ係数部分及び文字列部分がそれぞれ互いに素となるようにしてある。標準化を行なう前に因数分解を行なったり、共通因子を括り出したりする操作はない。

4．1　式の比較

２つの式が等しいかどうかは、それぞれ標準化された式を比較して調べる。等式Ａ＝ＢとＣ＝Ｄが等しいかどうかは以下の方法のいずれかで調べる。
1）　たすき掛け処理（数式Ａ＊Ｄ及びＢ＊Ｃを作り、標準化して比較する）
2）　引き算処理（数式Ａ－Ｂ及びＣ－Ｄを作り、標準化して比較する）
3）　ＡとＣが等しいときは数式ＢとＤを標準化して比較する
4）　２つの等式に共通する一次元変数についてそれぞれを解き、解かれた式の分母、分子をたすき掛け処理して２つの式を作り出し、それらを比較する。
比較の関数は、 a[cmp, [A, B], [C, D], Ans] である。比較を行なう過程において、どのようにして等式から等式への変形が行なわれたかを推定し、上述のいずれかの方法を選択している。従って、選択を間違えると比較が正しく行なえない。

4．2　等式の変形

式の変形を行なう関数を４つ作成した。例えば、

a[trans. 演算指示子. [A. B]. 式. [答左辺. 答右辺]]

ここで演算指示子は加減乗除のいずれかであり、A＝Bの両辺に与えられた式を加減乗除する。

a[trans. sol. [A. B]. 文字. [答左辺. 答右辺]]

指定した文字について等式を解く。

a[trans. rep. [A. B]. [文字. 式]. [答左辺. 答右辺]]

等式に含まれた文字を指定した式に置き換えて、その等式を作り直す。

5. むすび

電気回路演習の答案添削システムの概要を示し、添削に必要な回路知識の表現法について述べた。現在のシステムでは、オームの法則とキルヒホフの法則を使い、枝電流法、閉路電流法及び節点電位法のいずれかで解析した答案に対しては十分な知識を持つと考えられる。統括部の推論機能を増せば、混合解析も添削が可能である。実際に使用した時、以下の問題点が生じた。

1) 日本語構文解析部が十分でないために文章の意味を理解できない場合がある
2) 数式処理部が因数分解の機能を持たないために式の変形で記憶領域を越える場合が多い
3) 診断している文や式はその直前に意味情報が示されているものと考えているため、説明がない場合に対処できない
4) 式の変形では1段階推論を原則としているために、2段階以上の演算を同時に施して得られた式の比較には時間がかかり、答えが得られない場合がある。

これらの問題点はより良いプログラムを作成することによって解決できるものと考えられるが、現システムでは記憶容量の制限により大幅な改良は望めない。

6. 文献

(1) J. S. Brown. R. R. Burton and A. G. Bell. SOPHIE: A sophisticated instructional environment for teaching electronic troubleshooting. BBN Report No. 2790. Bolt Beranek and Newman. Cambridge Mass.. 1974

(2) R. M. Stallman and G. J. Sussman. " Problem solving about electrical circuits ". in Artificial Intelligence: An MIT Perspective. Vol. 1.. P. H. Winston and R. H. Brown. eds.. MIT Press. 1979

(3) 桑田他、" 電気回路演習を支援するエキスパートシステム(I). (II) "、情報処理学会第32回全国大会、pp1109-1112. 1986

(4) S. Watanabe. Y. Kuwata and T. Mizuta. " An expert system: An assistant to students in learning linear electrical network analysis". JSST Conference. pp198-203. 1986

7. 付録

```
Enter Sentence :
--->Sentence is [r1,wo,nagareru,denryuu,wo,i1,to,suru]
!!!! denryuu no muki ga sadamatpute inai
Enter Sentence :
--->Sentence is [i1,ha,node1,kara,node2,he,nagareru,to,suru]
--->denryuu ga teigi sareta
ok
Enter Sentence :
--->Sentence is [c,wo,node2,kara,node3,he,nagareru,denryuu,wo,i2,to,suru]
--->denryuu ga teigi sareta
ok
Enter Sentence :
--->Sentence is [i3,ha,1,wo,node2,kara,node3,he,nagareru,denryuu,to,suru]
--->denryuu ga teigi sareta
ok
```

```
Enter Sentence :
--->Sentence is [[r1,c,e],wo,loop1,to,suru]
--->loop ga teigi sareta
ok
Enter Sentence :
--->Sentence is [loop1,ni,kv1,wo,mochiiru,to,ex1,to,naru]
--->kv1 ga tekiou sareta.[r1 * i1 + (1 / (j * w * c) * i2 + (- e + 0)),0] ga tsu
kurareta
---> result o mushi shita
Enter Sentence :
--->Sentence is [ex1,e=r1*i1+i2/(j*w*c)]
--->[r1 * i1 + (1 / (j * w * c) * i2 + (- e + 0)),0] ??? [e,r1 * i1 + i2 / (j *
w * c)]
---> shiki ga teigi sareta. hikaku no ketsuka wa
ok
Enter Sentence :
--->Sentence is [[l,c],wo,loop2,to,katei,suru]
--->loop ga teigi sareta
ok
Enter Sentence :
--->Sentence is [loop2,ni,kv1,wo,tekiyousuru,to,ex2,j*w*l*i3-1/(j*w*c)*i2=0,to,n
aru]
--->kv1 ga tekiou sareta.[j * w * l * i3 + (- 1 / (j * w * c) * i2 + 0),0] ga ta
ukurareta
--->[j * w * l * i3 + (- 1 / (j * w * c) * i2 + 0),0] ??? [j * w * l * i3 - 1 /
(j * w * c) * i2,0]
---> shiki ga teigi sareta. hikaku no ketsu ka wa
ok
Enter Sentence :
--->Sentence is [node2,ni,kc1,wo,mochiiru,to,ex3,to,naru]
--->kc1 ga tekiou sareta.[i1 + (- i3 + (- i2 + 0)),0] ga tsukurareta
---> result o mushi shita
Enter Sentence :
--->Sentence is [ex3,i1=i2+i3]
--->[i1 + (- i3 + (- i2 + 0)),0] ??? [i1,i2 + i3]
---> shiki ga teigi sareta. hikaku no ketsuka wa
ok
Enter Sentence :
--->Sentence is [ex3,wo,ex1,ni,dainyuusuru]
--->trans ga yobareta[e,r1 * i1 + i2 / (j * w * c)][i1,i2 + i3]
--->ketsuka wa [e / 1,(c * i2 * r1 * w + c * i3 * r1 * w - i2 * j) / (c * w)]
---> dainyuu ketsuka wa [e / 1,(c * i2 * r1 * w + c * i3 * r1 * w - i2 * j) / (c
 * w)] de aru.
Enter Sentence :
--->Sentence is [ex4,e=r1*(i2+i3)+i2/(j*w*c)]
--->[e / 1,(c * i2 * r1 * w + c * i3 * r1 * w - i2 * j) / (c * w)] ??? [e,r1 * (
i2 + i3) + i2 / (j * w * c)]
---> shiki ga teigi sareta. hikaku no ketsuka wa
ok
Enter Sentence :
--->Sentence is [ex2,wo,i2,ni,tsuite,toku,to,ex5,i2=j*w*l*i3*(j*w*c),to,naru]
--->sol ga yobareta.[j * w * l * i3 - 1 / (j * w * c) * i2,0]i2
--->ketsuka wa[i2,(- c * i3 * l * w ^ 2) / 1] de aru.
--->[i2,(- c * i3 * l * w ^ 2) / 1] ??? [i2,j * w * l * i3 * (j * w * c)]
---> shiki ga teigi sareta. hikaku no ketsu ka wa
ok
Enter Sentence :
--->Sentence is [ex5,wo,ex4,ni,dainyuusuru,to,ex6,e=r1*(j^2*w^2*c*l*i3+i3)+j^2*w
^2*c*l*i3/(j*w*c),to,naru]
--->trans ga yobareta[e,r1 * (i2 + i3) + i2 / (j * w * c)][i2,j * w * l * i3 * (
j * w * c)]
--->ketsuka wa [e / 1,(- c * i3 * l * r1 * w ^ 2 + i3 * r1 + i3 * j * l * w) / 1
]
---> dainyuu ketsuka wa [e / 1,(- c * i3 * l * r1 * w ^ 2 + i3 * r1 + i3 * j * l
 * w) / 1] de aru.
--->[e / 1,(- c * i3 * l * r1 * w ^ 2 + i3 * r1 + i3 * j * l * w) / 1] ??? [e,r1
 * (j ^ 2 * w ^ 2 * c * l * i3 + i3) + j ^ 2 * w ^ 2 * c * l * i3 / (j * w * c)]
---> shiki ga teigi sareta. hikaku no ketsu ka wa
ok
Enter Sentence :
--->Sentence is [ex6,wo,i3,ni,tsuite,toku,to,ex,i3=e/(r1*j^2*w^2*c*l+r1+j*w*l),t
o,naru]
--->sol ga yobareta.[e,r1 * (j ^ 2 * w ^ 2 * c * l * i3 + i3) + j ^ 2 * w ^ 2 *
c * l * i3 / (j * w * c)]i3
--->ketsuka wa[i3,(- e) / (c * l * r1 * w ^ 2 - r1 - j * l * w)] de aru.
--->[i3,(- e) / (c * l * r1 * w ^ 2 - r1 - j * l * w)] ??? [i3,e / (r1 * j ^ 2 *
 w ^ 2 * c * l + r1 + j * w * l)]
---> shiki ga teigi sareta. hikaku no ketsu ka wa
ok
```