論 文

Technical Papers

論理文法「拡張DCSG」と回路理解への応用

A Logic Grammar Called Extended DCSG and Its Application to Circuit Understanding

田中 卓史 福岡工業大学
Takushi Tanaka Dept. of Computer Science and Engineering, Fukuoka Institute of Technology
tanaka@fit.ac.jp, http://www.fit.ac.jp/~tanaka/

keywords: logic programming, circuit grammar, understanding circuits, knowledge representation, generating sentences

Summary

First, we develop a type of logic grammar called Extended DCSG. This logic grammar not only defines the syntactic structures of free-word-order languages but also defines relationships between syntactic structures and their meanings. Next, we view electronic circuits as a free-word-order language, and write an Extended DCSG grammar for this language. Relationships between circuit structures and their functions are defined as grammar rules. The Extended DCSG grammar rules are converted into a logic program that parses electronic circuits and derives their functions and behaviors as meanings. These functions and behaviors are finally translated into natural language for explanation.

1. はじめに

句構造文法から語順の制約を取り去ると，語順を持たない言語を定義することができる．この語順を持たない言語を論理プログラミングの方法で生成・解析する確定節文法 DCSG(Definite Clause Set Grammar) が開発された [Tanaka 91]．DCSG は通常の確定節文法 DCG(Definite Clause Grammar) [Pereira 80] と類似の論理プログラミングの方法を用いたもので，文法規則を確定節に変換する際に DCG が差分リストを用いるところを，DCSG では差集合 (補集合) を用いている．

語順を持たない言語の下降解析は後向き推論の過程とよく似てくる．書き換え規則がルールに，対象となる文 (語の集合) がファクトに，出発記号がゴールに相当する．異なる点は下降解析ではどの語も出発記号への還元に一度しか使われないのに対して，後向き推論では同じファクトが推論に何度でも使われる点である．文中の各語が還元に一度しか使われない性質を利用すると，後向き推論において同じファクトを繰り返し使って無限ループに陥る問題を，無語順言語の構文解析の問題に置き換え，簡単に回避できるのである．すなわち，DCSG による下降解析はループの生じにくい後向き推論メカニズムとして使うことができる [Tanaka 91]．

DCSG は電子回路の構造解析に効果的に応用することができる．電子回路は一般に特定の目的のための機能ブロックとして設計される．機能ブロックはより下位の機能ブロックを組合せて構成され，最終的にトランジスタや抵抗器のような単純な機能の単一の部品から構成される．このような機能ブロックの階層構造が言語の文法構造に類似していることから，回路の構造を DCSG の文法規則として定義する方法が提案された [Tanaka 93]．回路の文法規則は DCSG コンバータ [Tanaka 99] により回路の構造解析を行うことのできる論理プログラムに変換される．

回路を文法的な文としてみると，回路の動作や機能などの電気的な性質は回路の構造から生じる意味として考えることができる．いま，文法と意味の関係をうまく定義できれば，与えられた回路構造から機能を求める解析や，逆に機能を実現する回路の設計に応用することができる．

そこで，この論文では初めに無語順言語の確定節文法 DCSG を拡張し，文法と意味の関係を定義できるようにする．次に，拡張した DCSG の有効性を確かめるために，論文 [Tanaka 93] で用いられた回路を例にとり，拡張 DCSG を用いて回路の構造と機能の関係を定義する．定義された規則は拡張 DCSG コンバータにより論理プログラムに変換される．得られた論理プログラムを用いて回路の構造解析を行うと，対象回路に関する多くの意味情報が導かれる．意味情報が導かれた状態は我々が回路を読み理解した状態に相当する．意味情報は論理命題の形で得られるが，用いられる項や述語に対して日本語のテンプレートを定義しているので，回路の動作や機能を日本語文として出力することもできる．

2. 無語順言語の確定節文法 DCSG

無語順言語の確定節文法 DCSG[Tanaka 91] は DCG と同じように終端記号や非終端記号が変数を持つことを許している．記号間で変数を共有することにより，語相互の複雑な制約を表すことができる．

句構造文法から語順を取り去ると，書き換え規則は語の集合を生成する．文の生成に用いられた非終端記号は，文 (単語の集合) の部分集合を生成したことになるので，書き換え規則はこれら部分集合の包含関係の定義として見ることができる．この包含関係を利用して，無語順言語の確定節文法 DCSG は次のように生成規則 (1) を Prolog 言語の節 (1)' に変換する．

$$A \longrightarrow B_1, B_2, ..., B_n. \quad (n \geq 1) \tag{1}$$

$$\begin{aligned} subset(A, S_0, S_n) :- \\ & subset(B_1, S_0, S_1), \\ & subset(B_2, S_1, S_2), \\ & \cdots , \\ & subset(B_n, S_{n-1}, S_n). \end{aligned} \tag{1)'}$$

述語 $subset(A, S_0, S_n)$ は A が集合 S_0 の部分集合で S_n がその補集合であることを述べている．この Prolog の節 (1)' は A が集合 S_0 の部分集合であり，S_n がその補集合であることを示すには，B_1 が集合 S_0 の部分集合であり，その補集合を S_1 として，B_2 が集合 S_1 の部分集合であり，...，B_n が集合 S_{n-1} の部分集合であり，その補集合が S_n となることを示せと読むことができる．

ここで，(1) の右辺 $B_1, B_2, ..., B_n$ は全て非終端記号を想定している．もし，B_i が終端記号の場合は非終端記号から区別するために $[B_i]$ のように"["と"]"で囲んでおく．終端記号は文 (語の集合) の要素であるから，Prolog の節に変換する際，述語 $subset$ の代わりに述語 $member$ を用いて，$member(B_i, S_{i-1}, S_i)$ に変換する．ここで述語 $member$ は (2) のように定義されている．すなわち，集合 S_{i-1} の中に要素 B_i が同定されると，その補集合が S_i から得られる．

$$\begin{aligned} & member(M, [M|X], X). \\ & member(M, [A|X], [A|Y]) :- \\ & \qquad member(M, X, Y). \end{aligned} \tag{2}$$

DCSG は下降解析を行うので，DCG と同じように左回帰の文法規則に出会うとループの問題を生じる．一般の文脈自由文法では補助的な非終端記号を導入することで，左回帰を右回帰に書き換えてループ問題を避けることができる [西田 81 (グライバッハの標準形)].

語順の無い言語では文法規則の右辺の順序も自由になるので左回帰を避けやすいが，回路文法において直並列回路 (2 端子素子の直列接続と並列接続だけで構成される回路) を定義しようとすると左右回帰となってループ問題を生じる．補助的な非終端記号として 3 端子回路を導入すると，右回帰の文法規則として直並列回路を定義することができる [田中卓史 87].

左回帰の問題を避ける別の方法として，文法規則を上昇解析の論理プログラムに変換する方法 BUP が開発されている [Matsumoto 83]．DCSG で行ったように，BUP が用いる差分リストを差集合で置き換えると，BUP の上昇解析の方法を無語順言語にも適用でき，補助的な非終端記号を導入することなく，直並列回路を定義することができる [Tanaka 91].

3. 意味項を持つ確定節文法「拡張 DCSG」

3·1 文法と意味により特徴づけられる言語

自然言語処理では構文解析や意味解析を通して，言語表現から格構造や概念依存構造などの意味表現が導かれる [田中穂積 99 (2 章)]．意味解析には概念や知識を蓄えた辞書が用いられる．自然言語は記述する世界が広いので語の数が多く，文法規則と比較して辞書は非常に大きなものとなる．一方，回路を言語のアナロジーでとらえると，自然言語とは逆に語の数が非常に少なく，文法規則として定義される構造の方が多くなる．また，回路の構造を文法に，機能を意味に対応させて考えるとき，構造と機能は不可分の関係にあり，文法と意味は一体として扱うのが効率が良い．

この節では DCSG の規則に新たに意味項を導入し，文法構造と意味の関係を直接的に定義できるように拡張する．文法が言語の表層を定義するのに対して，意味項は言語の表層から隠れた情報を扱う．意味項の使い方は言語の記述する世界と応用に依存する．文法が回路構造を定義すれば，意味項は表層に現われない電圧・電流と，それから生じる機能の定義に用いることができる．

DCSG では言語から語順を取り除いているので，通常，言語とは考えられないような単なるデータの集合でさえも，特定の世界を記述した文として見ることができる．例えば，ある回路上で測定した電圧データの集合を無語順言語の文として扱い，任意端子間の電圧を求める問題を構文解析の問題として解くことができる [Tanaka 91]．データ集合の中に意味のある構造が存在すれば非終端記号として定義すればよい．拡張 DCSG ではさらにその構造の意味も同時に定義できるようにしている．すなわち，通常の文法と異なり，拡張 DCSG は意味も含んだ形で言語を特徴づけることになる．

拡張 DCSG では，意味項を書き換え規則の左辺と右辺に書くことができ，それぞれ働きが異なる．また，語の意味も文法構造により生じる意味と同じ形式で統一的に扱うことができるように，書き換え規則における終端記号の表現と取り扱いを改めている．

3·2 左辺の意味項

書き換え規則の左辺の意味項は右辺の文法構造から生じる意味を定義するもので，(3) に示すように “{” と “}” で囲まれる．

$$A,\{F_1,F_2,...,F_m\} \longrightarrow B_1,B_2,...,B_n. \qquad (3)$$

この規則は意味 $\{F_1,F_2,...,F_m\}$ を持つ記号 A が文法構造 $B_1,B_2,...,B_n$ から構成されると読む．拡張 DCSG ではこの形の規則を次の確定節に変換する．確定節への変換は DCSG の場合と異なるので，述語 *subset* の代わりに述語 *ss* を用いている．

$$\begin{aligned} ss(A,S_0,S_n,E_0,[F_1,F_2,...,F_m|E_n]) :- \\ & ss(B_1,S_0,S_1,E_0,E_1), \\ & ss(B_2,S_1,S_2,E_1,E_2), \\ & \cdots, \\ & ss(B_n,S_{n-1},S_n,E_{n-1},E_n). \end{aligned} \qquad (3)'$$

この規則を無語順言語の構文解析に使う場合，変数 S_0 に対象集合を，変数 E_0 には空集合を代入したゴール節 $ss(A,S_0,S_n,E_0,E)$ を実行する．対象集合 S_0 中に部分集合 “$B_1,B_2,...,B_n$”が順次同定され，S_n からは対象集合から A を除いた集合 (補集合) が得られる．一方，E_0 に部分集合 B_1 を同定することで得られた意味情報が付加されて E_1 となり，さらに，B_2 を同定することで得られた意味情報が付加されて E_2 となり，...，さらに，B_n を同定することで得られた意味情報が付加され E_n となる．右辺の部分集合 “$B_1,B_2,...,B_n$”がすべて同定されると，構造 A が同定されたとして，E_n にさらに意味情報 “$F_1,F_2,...,F_m$”が付加されて，非終端記号 A に関する意味情報が変数 E から得られる．

変数の列 “$S_0,S_1,...,S_n$”には構造解析の過程における対象集合のまだ同定されていない部分が蓄えらているので，次第に要素の数が減少する．一方，変数の列 “$E_0,E_1,...,E_n$”には構造解析の過程で明らかになる意味情報が蓄えられるので，次第に要素の数が増加する．すなわち，これら二つの変数の列は文の構造が明らかになるにつれて，文の構成要素の未知の部分が減少し，一方，文の意味が次第に明らかになって行く様子を表している．

3·3 右辺の意味項

書き換え規則の右辺の意味項は文法規則を用いて構文解析を行う場合の意味的制約条件を表すもので，回路の文脈依存構造の解析 [Tanaka 93] に利用することができる．意味項は左辺の場合と同じように “{” と “}” で囲まれ，右辺の任意の位置に複数個現れてよい．例えば，次の形の規則 (4) は拡張 DCSG コンバータにより，(4)’ に変換される．

$$A \longrightarrow B_1,\{C_1,C_2\},B_2. \qquad (4)$$

$$\begin{aligned} ss(A,S_0,S_n,E_0,E_n) :- \\ & ss(B_1,S_0,S_1,E_0,E_1), \\ & member(C_1,E_1,\_), \\ & member(C_2,E_1,\_), \\ & ss(B_2,S_1,S_2,E_1,E_2). \end{aligned} \qquad (4)'$$

(4)’ が無語順言語の解析に使われる場合，非終端記号 A を同定するため，記号 B_1 が同定された後，意味条件 C_1,C_2 がそれまでに得られた意味情報 E_1 中に存在するかどうか調べられる．意味条件が満足されると，次の記号 B_2 の同定に進む．すなわち，この形の意味項は意味条件のテストに用いられるだけで，解析により得られた意味情報を増減させるようなことはない．

3·4 終端記号と意味項

無語順言語の確定節文法 DCSG では文法規則中の終端記号は “[” と “]” で囲まれて区別されたが，新たに拡張した DCSG ではこのような表記上の区別を行わない．したがって，終端記号も非終端記号も述語 *ss* を用いて確定節に変換される．その代わりに，終端記号はこれ以上書き換えることができないので，書き換え規則の左辺だけの規則として定義する．いま，終端記号 A が意味 $\{F_1,F_2,...,F_m\}$ を持つならば，(5) のように A の後にカンマで区切って書いておく．

$$A,\{F_1,F_2,...,F_m\}. \qquad (5)$$

これは拡張 DCSG コンバータにより，次の確定節 (5)’ に変換される．

$$\begin{aligned} ss(A,S_0,S_1,E_0,[F_1,F_2,...,F_m|E_0]) :- \\ & member(A,S_0,S_1). \end{aligned} \qquad (5)'$$

すなわち，構文解析の際に，終端記号 A に対して (5)’ の節が適用されると，まず，記号 A が現在の解析対象の集合 S_0 中に要素として存在するかどうか調べられ，存在すればその補集合が S_1 に返される．また，現在までに構文解析により得られた意味情報 E_0 に，新たにその終端記号の意味 $F_1,F_2,...,F_m$ が付加されて，述語 *ss* の第 5 引数から戻される．こうすることで，別に単語の意味辞書のようなものを用意することなく，文法構造により生じる意味と統一的な形で，語の意味を扱うことができる．

4. 回路の知識表現

電子回路の教科書では回路知識が言語だけではなく数式や信号波形，特性曲線などを用いて表されているので，必ずしもすべて記号的な方法で回路の意味が表現できるとは考えていない．しかし，数式や図形も言語で述べられる知識が基本に存在すると考えている．この節では，回路が機能を発揮するときの電圧・電流の状態や依存関係，回路の機能や目的など言語的に表現しやすい知識に

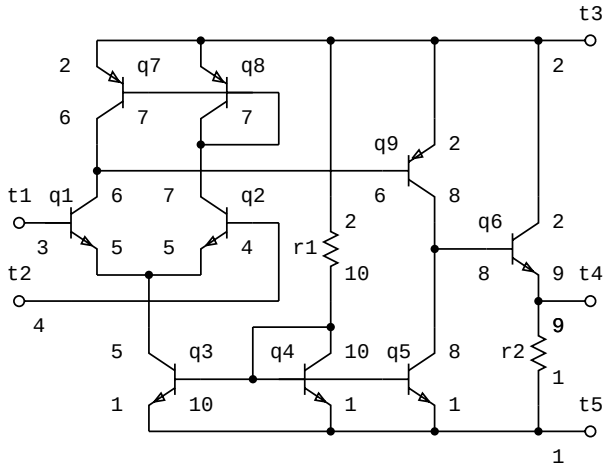

図 1 オペレーショナル・アンプ *cd*42

着目し，拡張 DCSG を用いて，回路の構造と意味の関係を定義する．

4·1 回路自体の表現

電子回路の構造解析の研究 [Tanaka 93] において図 1 の回路 [IC 76] は Prolog のリスト (6) を用いて表される．リストの要素 $resistor(r1,2,10)$ は節点 2 と節点 10 につながる抵抗器 r1 を，$npnTr(q1,3,5,6)$ はベースが節点 3 に，エミッタが節点 5 に，コレクタが節点 6 につながる NPN トランジスタ q1 を，$terminal(t1,3)$ は節点 3 につながる外部端子 $t1$ を表している．これら各素子に対応する複合項が回路を記述する文の単語となる．リスト (6) は無語順言語の文として単語の集合の表現に用いられているので，項の順序に意味はない．

$$\begin{aligned}&[resistor(r1,2,10), resistor(r2,9,1),\\ &\ npnTr(q1,3,5,6), npnTr(q2,4,5,7),\\ &\ npnTr(q3,10,1,5), npnTr(q4,10,1,10),\\ &\ npnTr(q5,10,1,8), npnTr(q6,8,9,2),\\ &\ pnpTr(q7,7,2,6), pnpTr(q8,7,2,7),\\ &\ pnpTr(q9,6,2,8), terminal(t1,3),\\ &\ terminal(t2,4), terminal(t3,2),\\ &\ terminal(t4,9), terminal(t5,1)]\end{aligned} \tag{6}$$

4·2 回路の動作や機能の表現

回路の動作や機能などは回路の構造から生じる意味と考えている．回路の意味も回路自体の表現と同じように複合項の集合で表すことにする．回路もその意味もどちらも複合項の集合となるが，回路表現は回路文法で定義された文法的な文として見なされるのに対して，意味表現は独自の生成規則を持たず，回路の構造と意味の定義から構造解析の過程で生成される．

回路を構成する素子の種類は限られているので，おのずと回路表現に必要な語は定まったのであるが，回路の動作や機能などの記述に必要となる語はあらかじめ定めがたく，文法規則に意味情報を付加する過程で追加して行くことになる．そこで，意味表現が人に正しく理解できることが必要になるので，新たな語 (項や関数，述語) を用いるときは対応する日本語のテンプレートを定義し，日本語としても読めるようにする (第 6 節)．

例えば，節点 A，B 間の電圧は $voltage(A,B)$ のように表し，節点 A から素子 X への電流 (枝電流) は $current(A,X)$ のように表して，次の名詞句テンプレート $tj(Term, Japanese)$ を定義する．

$$\begin{aligned}&tj(voltage(A,B), [節点, A, ', ', B, 間の電圧]).\\ &tj(current(A,X), [節点, A, から, X, への電流]) :-\\ &\qquad number(A).\\ &tj(current(X,A), [X, から, 節点, A, への電流]) :-\\ &\qquad number(A).\end{aligned}$$

4·3 終端記号に付加される意味項

回路の構造解析 [Tanaka 93] では回路の終端記号として外部端子，抵抗器，バイポーラトランジスタが用いられている．論理文法 DCSG では文法規則中の終端記号は “[” と “]” で囲まれたが，前節で定義した新たな拡張 DCSG では非終端記号と終端記号に表記上の区別はなく，終端記号は右辺を持たない規則として定義する．例えば外部端子は次のように左辺だけの規則となる．

$$terminal(A,N). \tag{7}$$

抵抗器が終端記号であることを示すには左辺だけの規則 (8) を定義する．

$$resistor(R,A,B). \tag{8}$$

回路中に抵抗器 $resistor(R,A,B)$ が同定されると，抵抗器の端子間電圧 $voltage(A,B)$，抵抗器の枝電流 $current(A,R)$ などを考えることができる．抵抗器の電圧と電流はオームの法則の制約を受ける．この制約は数式を用いて正確に表すこともできるが，この研究ではより基本的な知識として，電気的な事象 (電圧，電流) の依存関係に着目している．物理現象における依存関係は一般に因果関係と呼ばれることが多い．抵抗器における電圧と電流の依存関係は物理的にどちらが原因でどちらが結果とも言い難いが,「電圧が加わることで電流が流れた」，あるいは「電流が流れることで電圧が発生した」のように，原因と結果の関係として見ることで回路の動作を定性的に把握することが可能になる．このような定性的な原因・結果の把握は回路の動作理解や故障診断*1の

*1 アナログ・デジタルを問わず回路の故障診断は一般に電源電圧が正しく供給されているかを調べ，次ぎに信号の流れを入力または出力から辿り故障箇所を見つける．電源電圧の供給は回路上のすべての因果関係の前提条件となっている．素子や機能ブロック自体の故障は因果連鎖の切断箇所として特定できる．一方，プリント板上の半田ブリッジによる故障は回路自体が別の回路へと変化するため異常な因果関係を示し，回路図上からの故障箇所の特定はより難しくなる．

基本となる．

そこで，終端記号としての抵抗器の定義 (8) の代わりに，これらの電圧・電流の因果関係を意味項として加えた (8)' を考える．

$$resistor(R,A,B),\\ \{cause(voltage(A,B),current(A,R)),\\ cause(current(A,R),voltage(A,B))\}. \qquad (8)'$$

ここで因果関係の定義には述語 $cause$ を用いている．2 つの $cause$ の項は抵抗器の電圧・電流が相互に他方の原因となることを表している．なお，抵抗器のように極性を持たない素子の場合，このような電圧・電流の因果関係は (8)' の形で定義するよりも (8) のままにして，極性を持たない素子のための規則 (12) で定義する方が優れている (後述)．

トランジスタの場合は 3 つの動作状態 (能動，飽和，遮断) で電圧・電流の因果関係が異なるので，意味項の異なる同じ終端記号を 3 通り定義することになる．(9) に能動状態の NPN トランジスタの定義を示している．この規則が適用されると，そのトランジスタ Q が能動状態にあることを示す $state(Q,active)$ が意味情報として得られる．また，能動状態における端子間電圧や枝電流の大小関係，電圧・電流間の因果関係も得られる．トランジスタの意味項は電流増幅率などを加えてより詳細なものにすることもできる．同様に意味項だけが異なる終端記号として，飽和状態や遮断状態のトランジスタを定義する．

$$npnTr(Q,B,E,C),\\ \{state(Q,active),\\ gt(voltage(C,E),vst),\\ equ(voltage(B,E),vbe),\\ gt(current(B,Q),0),\\ gt(current(C,Q),0),\\ cause(voltage(B,E),current(B,Q)),\\ cause(current(B,Q),current(C,Q))\}. \qquad (9)$$

$$pnpTr(Q,B,E,C),\\ \{state(Q,active),\\ gt(voltage(E,C),vst),\\ equ(voltage(E,B),vbe),\\ gt(current(Q,B),0),\\ gt(current(Q,C),0),\\ cause(voltage(E,B),current(Q,B)),\\ cause(current(Q,B),current(Q,C))\}. \qquad (10)$$

これまでに用いた項や述語に対して，次ぎのように名詞句テンプレート $tj(Term,Japanese)$，および述語テンプレート $pj(Predicate,Japanese)$ を定義する．

$tj(active,$能動$)$.
$tj(vst,$コレクタ飽和電圧$)$.
$tj(vbe,$ベースエミッタ間順電圧$)$.
$pj(cause(C,E),[C,$は$,E,$の原因となる$])$.
$pj(state(Q,S),[Q,$は$,S,$状態にある$])$.
$pj(gt(X,V),[X,$は$,V,$より大きい$])$.
$pj(equ(X,V),[X,$は$,V,$に等しい$])$.

4·4 構造定義に付加される意味項

規則 (11) はダイオード接続トランジスタ (図 2) の構造定義に導通状態の電圧や電流，因果関係などの意味情報を加えたものである．規則の左辺の $dtr(dtr(Q),A,C)$ はアノードが節点 A に，カソードが節点 C に接続されたダイオード接続トランジスタを表している．$dtr(Q)$ はこのダイオード接続トランジスタに与えた名前 (スコーレム関数) である．左辺の意味項はこのダイオード接続トランジスタが導通状態 $state(dtr(Q),conductive)$ にあるとき，アノード A から $dtr(Q)$ に電流が流入すること，流入する電流 $current(A,dtr(Q))$ が両端に発生する電圧 $voltage(A,C)$ の原因となることを定義している．これらの電圧・電流は抵抗器の場合と同じようにどちらが原因でどちらが結果とも言えないが，実際の回路では電流で電圧を制御するような使い方しか行われないので，一方向の定義となっている．

右辺の ";" (論理和) で結合された項 $npnTr(Q,A,C,A)$ と $pnpTr(Q,C,A,C)$ はそれぞれ図 2 の (a) と (b) の構造に対応し，どちらかが存在すればよいことを表している．右辺の意味項 $\{state(Q,active)\}$ はこのトランジスタ Q が電気的に能動状態で働いていることを要求している．すなわち，図 2 の接続条件を満たすトランジスタが回路中に同定されると，そのトランジスタに対して意味条件として，遮断状態でも，飽和状態でもなく，能動状態の定義を用いることを要求している．この能動状態の定義から節点 A, C 間の電圧が vbe(=0.6〜0.7v) であることなどが導かれる．この右辺の接続条件と電気的条件により，導通状態のダイオード接続トランジスタ $dtr(dtr(Q),A,C)$ が定義される．同じようにして意味項だけ異なる遮断状態の規則も定義する．

$$dtr(dtr(Q),A,C),\\ \{state(dtr(Q),conductive),\\ gt(current(A,dtr(Q)),0),\\ cause(current(A,dtr(Q)),voltage(A,C))\} \longrightarrow\\ (npnTr(Q,A,C,A);\ pnpTr(Q,C,A,C)),\\ \{state(Q,active)\}. \qquad (11)$$

端子の順番に無関係に抵抗器を指示するための規則に意味項として抵抗器の電気的性質を加えておくことができる (12)．この場合，終端記号としての抵抗器の定義は規則 (8) を用いる．今，規則 (8)' を使うと，たまたま入力された回路の抵抗器の向き (節点の順番) が電圧・電流の正の向きとして使われるが，規則 (12) を使うと，文法規則中で言及される抵抗器の向きが正の向きとして使わ

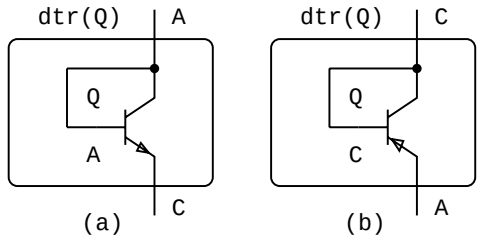

図 2 ダイオード接続トランジスタ

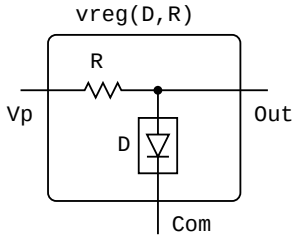

図 3 電圧レギュレータ

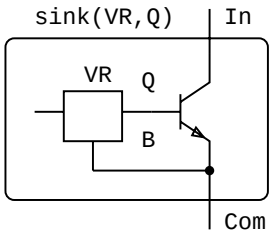

図 4 電流源

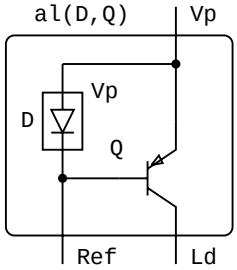

図 5 アクティブロード

れて理解しやすくなる．

$$
\begin{aligned}
&res(R,A,B),\\
&\{cause(voltage(A,B),current(A,R)),\\
&\ cause(current(A,R),voltage(A,B))\} \longrightarrow\\
&\qquad (\ resistor(R,A,B);\\
&\qquad\ \ resistor(R,B,A)\ ).
\end{aligned}
\tag{12}
$$

規則 (13) は図 3 の簡単な電圧レギュレータを定義している．左辺の意味項はこの電圧レギュレータ $vreg(D,R)$ が節点 Out，Com 間の電圧を制御することを表している．右辺は図 3 の構造定義の他に意味項 $\{state(D,conductive)\}$ を持っている．この意味項はダイオード接続トランジスタが電気的に導通状態にあることを要求しており，導通状態の定義 (11) が使われることになる．

$$
\begin{aligned}
&vbeReg(vreg(D,R),Vp,Com,Out),\\
&\{control(vreg(D,R),voltage(Out,Com))\} \longrightarrow\\
&\qquad dtr(D,Out,Com),\\
&\qquad \{state(D,conductive)\},\\
&\qquad res(R,Vp,Out).
\end{aligned}
\tag{13}
$$

規則 (14) は図 4 の電流源を定義している．左辺の意味項はこの電流源 $sink(VR,Q)$ が節点 In から Q へ流入する電流を制御することを表している．右辺の最初の項は回路の構造条件と意味条件が論理和 ";"で結ばれている．すなわち，構造条件として電圧レギュレータ VR が存在するか，または，現在の意味情報の中に節点 B と Com 間の電圧が回路 VR により制御されることを述べた項があればよいことを表しており，文脈依存型回路の解析 [Tanaka 93] において効果を発揮する．右辺の残りの部分は電流源の本体となる NPN トランジスタ Q が存在すること，及び，そのトランジスタが能動状態にあることを意味的条件として要求している．

$$
\begin{aligned}
&cSink(sink(VR,Q),In,Com),\\
&\{control(sink(VR,Q),current(In,Q))\} \longrightarrow\\
&\qquad (\ vbeReg(VR,\_,Com,B);\\
&\qquad\ \ \{control(VR,voltage(B,Com))\}\ ),\\
&\qquad\ npnTr(Q,B,Com,In),\\
&\qquad\ \{state(Q,active)\}.
\end{aligned}
\tag{14}
$$

規則 (15) は図 5 に示すアクティブ・ロード (カレントミラー) を定義している．左辺の意味項として，この回路 $al(D,Q)$ がトランジスタ Q から節点 Ld への電流を制御すること，アクティブ・ロードから節点 Ref への電流が Q から節点 Ld への電流の原因となること，Q から節点 Ld への電流はアクティブ・ロードから節点 Ref への電流に等しいことを表している．一方，右辺は図 5 の構造定義に対して，さらに意味条件として，ダイオード接続トランジスタ D が導通状態にあること，トランジスタ Q が能動状態にあることを要求している．

$$
\begin{aligned}
&activeLoad(al(D,Q),Ref,Vp,Ld),\\
&\{control(al(D,Q),current(Q,Ld)),\\
&\ cause(current(al(D,Q),Ref),current(Q,Ld)),\\
&\ equ(current(Q,Ld),current(al(D,Q),Ref))\}\\
&\qquad\qquad \longrightarrow\\
&\qquad\qquad dtr(D,Vp,Ref),\\
&\qquad\qquad \{state(D,conductive)\},\\
&\qquad\qquad pnpTr(Q,Ref,Vp,Ld),\\
&\qquad\qquad \{state(Q,active)\}.
\end{aligned}
\tag{15}
$$

規則 (16) は図 6 の差動増幅器を定義している．規則 (16) の左辺は意味項として回路の入出力となる電圧・電流，及びそれらの因果関係，回路を構成するブロックの回路内での機能を定義している．右辺は図 6 の構造を表

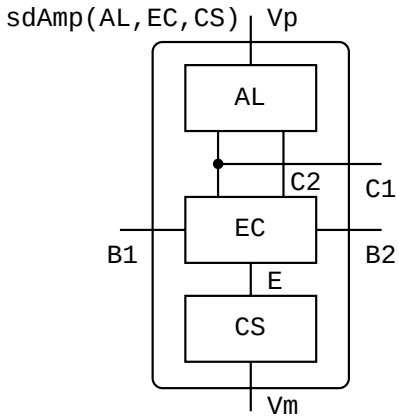

図 6 差動増幅器

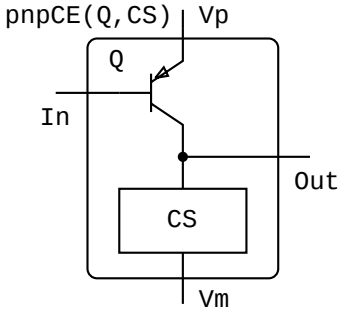

図 7 コモンエミッタ

す接続条件だけから構成されている．

$$
\begin{aligned}
&sDiffAmp(sdAmp(EC,AL,CS),\\
&\qquad B1,B2,C1,Vp,Vm),\\
&\{input(sdAmp(EC,AL,CS),\\
&\quad difference(voltage(B1,Vm),\\
&\qquad voltage(B2,Vm))),\\
&\ output(sdAmp(EC,AL,CS),\\
&\quad current(C1,sdAmp(EC,AL,CS))),\\
&\ cause(voltage(B1,B2),\\
&\quad current(C1,sdAmp(EC,AL,CS))),\\
&\ supress(CS,\\
&\quad common\_mode\_gain(sdAmp(EC,AL,CS))),\\
&\ double(AL,current\_gain(EC))\} \longrightarrow\\
&\quad eCoupledPair(EC,B1,B2,E,C1,C2),\\
&\quad activeLoad(AL,C2,Vp,C1),\\
&\quad cSink(CS,E,Vm). \qquad (16)
\end{aligned}
$$

規則 (17) は図 7 のコモンエミッタ (エミッタ接地) 回路を定義している．左辺の最初の意味項はこの回路に高インピーダンス負荷が接続された場合を想定している．低インピーダンス負荷が接続された場合の定義は別になる．次にベース電流が入力となること，コレクタと節点 Vm 間の電圧が出力となること，これらの電流，電圧が因果関係をなすこと，電圧利得が高いことを加えている．右辺は回路の構造がトランジスタ Q と電流シンク CS から

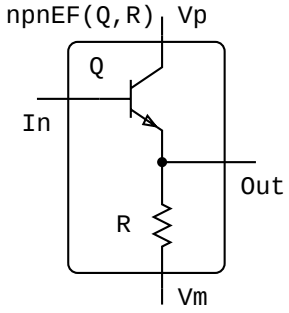

図 8 エミッタフォロワ

なることを表している．意味項はトランジスタ Q が能動状態にあることを要求している．

$$
\begin{aligned}
&commonEmitter(pnpCE(Q,CS),\\
&\qquad In,Out,Vp,Vm),\\
&\{high(load\_impedance(pnpCE(Q,CS))),\\
&\ input(pnpCE(Q,CS),current(Q,In)),\\
&\ output(pnpCE(Q,CS),voltage(Out,Vm)),\\
&\ cause(current(Q,In),voltage(Out,Vm)),\\
&\ low(input\_impedance(pnpCE(Q,CS))),\\
&\ high(output\_impedance(pnpCE(Q,CS))),\\
&\ high(voltage\_gain(pnpCE(Q,CS)))\} \longrightarrow\\
&\quad pnpTr(Q,In,Vp,Out),\\
&\quad \{state(Q,active)\},\\
&\quad cSink(CS,Out,Vm). \qquad (17)
\end{aligned}
$$

規則 (18) はエミッタフォロワー回路 (図 8) を定義している．左辺の意味項として，節点 In と Vm 間に電圧が入力されること，節点 Out と Vm 間に電圧が出力されること，これらの電圧が因果関係をなすこと，入力インピーダンスが高いこと，出力インピーダンスが低いこと，電圧利得が 1 であることを定義している．右辺の意味項はトランジスタ Q が能動状態にあることを要求している．

$$
\begin{aligned}
&emitterFollower(npnEF(Q,R),\\
&\quad In,Out,Vp,Vm),\\
&\{input(npnEF(Q,R),voltage(In,Vm)),\\
&\ output(npnEF(Q,R),voltage(Out,Vm)),\\
&\ cause(voltage(In,Vm),voltage(Out,Vm)),\\
&\ high(input\_impedance(npnEF(Q,R))),\\
&\ low(output\_impedance(npnEF(Q,R))),\\
&\ equ(voltage\_gain(npnEF(Q,R)),1)\} \longrightarrow\\
&\quad npnTr(Q,In,Out,Vp),\\
&\quad \{state(Q,active)\},\\
&\quad res(R,Out,Vm). \qquad (18)
\end{aligned}
$$

規則 (19) は図 9 の外部端子を持つオペレーショナル・アンプを定義したもので，入出力となる電圧，電圧の因果関係，構成要素である機能ブロックの役割を意味項に定義している．

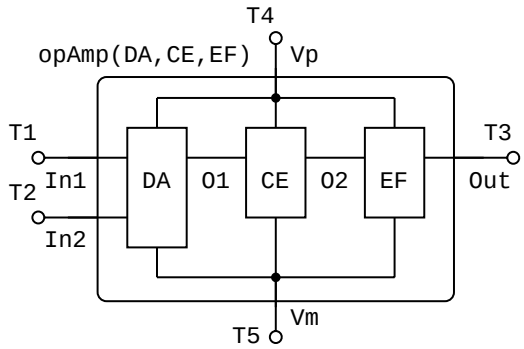

図 9 オペレーショナル・アンプ

$$
\begin{aligned}
&opAmp(opAmp(DA,CE,EF),\\
&\qquad In1,In2,Out,Vp,Vm),\\
&\{input(opAmp(DA,CE,EF),voltage(In1,In2)),\\
&\ output(opAmp(DA,CE,EF),voltage(Out,Vm)),\\
&\ cause(voltage(In1,In2),voltage(Out,Vm)),\\
&\ enable(DA,amplify(opAmp(DA,CE,EF),\\
&\qquad differential\_inputs)),\\
&\ enable(CE,\\
&\quad high(voltage\_gain(opAmp(DA,CE,EF)))),\\
&\ enable(EF,\\
&\quad low(output\_impedance(opAmp(DA,\\
&\qquad CE,EF))))\} \longrightarrow\\
&\qquad sDiffAmp(DA,In1,In2,O1,Vp,Vm),\\
&\qquad commonEmitter(CE,O1,O2,Vp,Vm),\\
&\qquad emitterFollower(EF,O2,Out,Vp,Vm),\\
&\qquad terminal(T1,In1),\\
&\qquad terminal(T2,In2),\\
&\qquad terminal(T3,Out),\\
&\qquad terminal(T4,Vp),\\
&\qquad terminal(T5,Vm). \qquad (19)
\end{aligned}
$$

新たな用語に対して，次の日本語のテンプレートを定義する．

$tj(conductive,$導通$).$
$tj(constant_current,$一定の電流$).$
$tj(differential_inputs,$差分入力$).$
$tj(common_mode_gain(X),[X,$の同相利得$]).$
$tj(current_gain(X),[X,$の電流利得$]).$
$tj(voltage_gain(X),[X,$の電圧利得$]).$
$tj(input_impedance(X),[X,$の入力インピーダンス$]).$
$tj(output_impedance(X),[X,$の出力インピーダンス$]).$
$tj(load_impedance(X),[X,$の負荷インピーダンス$]).$
$tj(difference(X,Y),[X,$と$,Y,$の差$]).$
$pj(high(X),[X,$は$,$高い$]).$
$pj(low(X),[X,$は$,$低い$]).$
$pj(amplify(X,Y),[X,$は$,Y,$を増幅する$]).$
$pj(control(X,Y),[X,$は$,Y,$を制御する$]).$
$pj(input(X,Y),[X,$は$,Y,$が入力される$]).$

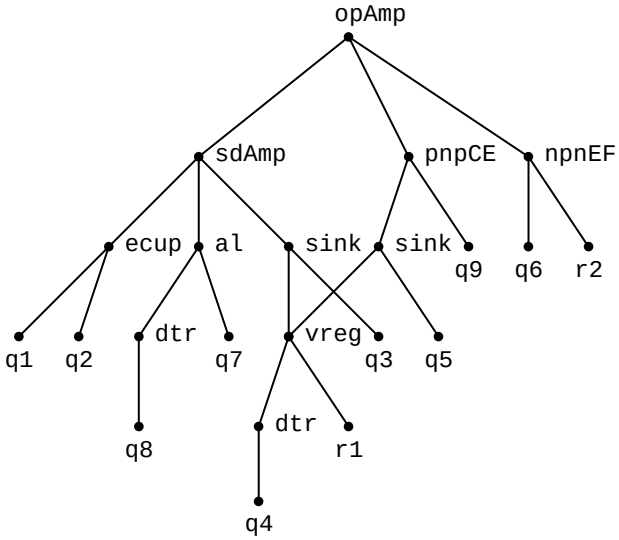

図 10 回路 $cd42$ の解析木

$pj(output(X,Y),[X,$は$,Y,$を出力する$]).$
$pj(supress(X,Y),[X,$は$,Y,$を抑制する$]).$
$pj(double(X,Y),[X,$は$,Y,$を 2 倍にする$]).$
$pj(cause(C,E),[C,$は$,E,$の原因となる$]).$
$pj(enable(X,Y),[X,$は$,Y,$を可能にする$]).$

5. 意味情報の導出

前節で定義した文法規則を拡張 DCSG コンバータに通すと下降解析を実行できる Prolog のプログラムが得られる．このプログラムを用いて次のゴール節 (20) を実行すると，変数 X からは回路 $cd42$ の全体を表す複合項が，変数 Y からは構造解析の過程で集められた意味情報が得られる．

回路全体を表す複合項 (21) の第一引数はその回路に与えた名前である．この名前は下降解析の過程で成功したサブゴールが記録されており，図 10 の構造解析木として表すことができる．解析木のノードは対象回路中に同定された機能ブロックに対応している．各ノードは意味情報の中から複合項により言及される．回路の構成規則をすべて文脈自由文法で表すことができれば回路の構造は完全な木の形となるが，文脈に依存する構造が含まれていると，図 10 のように木の途中が接続された構造を生じる [Tanaka 93].

変数 Y からは 103 個の意味情報が得られている (20 個表示)．各要素は同定された機能ブロック (解析木のノード) を指示するための複合項を含んでいる．これらの複合項は長くなるので省略形を用いて印刷している．

$$
\begin{aligned}
&?- \ cd42(Circuit),\\
&\qquad ss(X,Circuit,[\,],[\,],Y). \qquad (20)
\end{aligned}
$$

$X = opAmp(opAmp($

$$
\begin{aligned}
&sdAmp(ecup(q1,q2),\\
&\quad al(pdtr(q8),q7),\\
&\quad sink(vreg(dtr(q4),r1),q3)),\\
&pnpCE(q9,\\
&\quad sink(vreg(dtr(q4),r1),q5)),\\
&npnEF(q6,r2)),\\
&3,4,9,2,1)
\end{aligned}
\tag{21}
$$

$$
\begin{aligned}
Y = [&input(opAmp(...),voltage(3,4)),\\
&output(opAmp(...),voltage(9,1)),\\
&cause(voltage(3,4),voltage(9,1)),\\
&enable(sdAmp(...),\\
&\quad amplify(opAmp(...),differential\_inputs)),\\
&enable(pnpCE(...),\\
&\quad high(voltage\_gain(opAmp(...)))),\\
&enable(npnEF(...),\\
&\quad low(output\_impedance(opAmp(...)))),\\
&input(npnEF(...),voltage(8,1)),\\
&output(npnEF(...),voltage(9,1)),\\
&cause(voltage(8,1),voltage(9,1)),\\
&high(input\_impedance(npnEF(...))),\\
&low(output\_impedance(npnEF(...))),\\
&equ(voltage\_gain(npnEF(...)),1),\\
&cause(voltage(9,1),current(9,r2)),\\
&cause(current(9,r2),voltage(9,1)),\\
&state(q6,active),\\
&gt(voltage(2,9),vst),\\
&equ(voltage(8,9),vbe),\\
&gt(current(8,q6),0),\\
&gt(current(2,q6),0),\\
&cause(voltage(8,9),current(8,q6)),\\
&...\ ]
\end{aligned}
\tag{22}
$$

6. 意味情報の日本語表示

構造解析により得られた多くの意味情報を理解しやすいように日本語文へ変換する．

6·1 回路上の指示対象

与えられた回路は素子と節点から構成されている．回路図に示された素子や端子の名前 $q1, q2, ..., t1, t2, ...$ は固有名詞と考え，日本語文においてもそのまま用いる．一方，節点は数字で表されているので，日本語文では数字の前に「節点」を付ける．

構造解析により回路上に機能ブロックが同定されると，新たな指示対象として関数により名前が与えられる．この名前は次々に組み合わされて，図 10 に示すような木構造をなす．木のノードを構成する複合項を固有名詞としてそのまま機能ブロックの指示に用いることもできるが，日本語文の中では適切な名詞句に変換した方が読みやすくなる．

機能ブロック名を構成する関数は回路文法を定義した時点で次の名詞句テンプレートが想定されている．

$tj0(dtr(Q),[Q,$のダイオード接続$])$.
$tj0(ecup(Q1,Q2),[Q1,$と$,Q2,$のエミッタ結合ペア$])$.
$tj0(vreg(D,R),[D,$と$,R,$からなる電圧レギュレータ$])$.
$tj0(sink(VR,Q),[VR,$と$,Q,$からなる電流シンク$])$.
$tj0(al(D,Q),[D,$と$,Q,$からなるアクティブロード$])$.
$tj0(sdAmp(EC,AL,CS),$
$\quad[EC,$と$,AL,$’，’$,CS,$からなる差動増幅器$])$.
$tj0(pnpCE(Q,CS),$
$\quad[Q,$と$,CS,$からなるコモンエミッタ$])$.
$tj0(npnEF(Q,R),$
$\quad[Q,$と$,R,$からなるエミッタフォロワ$])$.
$tj0(opAmp(DA,CE,EF),$
$\quad[DA,$と$,CE,$’，’$,EF,$
$\qquad$からなるオペレーショナルアンプ$])$.

このテンプレートにより複合項 $dtr(q4)$ は「q4 のダイオード接続」となり，$al(dtr(q8),q7)$ は「q8 のダイオード接続と q7 からなるアクティブロード」となる．しかし，日本語に直接，変換して無理なく読めるのはこの程度の構造までで，入れ子がさらに増えた $pnpCE(q9,sink(vreg(dtr(q4),r1),q5))$ では「q9 と q4 のダイオード接続と r1 からなる電圧レギュレータと q5 からなる電流シンクからなるコモンエミッタ」となり，係り受けの曖昧さが増え，正しく読むことが困難になる．すなわち，対象の指示に便利な複合項も，入れ子が多いと構造を保存したまま名詞句に変換するのは適切でない．

今，ユーザインターフェースにより図 10 のように解析木が与えられているとする．回路上の機能ブロックは構造解析木のノードに限られるので,「アクティブロード」と言えばユニークに機能ブロック $al(dtr(q8),q7)$ が定まる．一方，ダイオード接続トランジスタや電流シンクのように複数個存在するものは「$q4$ のダイオード接続」や「$q3$ を含む電流シンク」のように，区別に必要なだけの終端記号を含む名詞句を作ればよい．そこで，同じ関数名のノードが他にない場合は機能ブロック名だけで対象を指示し，そうでない場合は $tj0$ を簡略化したテンプレート $tj1$ を用いると，簡潔かつユニークに対象を指示する名詞句が得られる．

$tj1(vreg(dtr(Q)),_),[Q,$を含む電圧レギュレータ$])$.
$tj1(sink(_,Q),[Q,$を含む電流シンク$])$.
$tj1(al(_,Q),[Q,$を含むアクティブロード$])$.
$tj1(sdAmp(ecup(Q1,Q2),\_,\_),$
$\quad[Q1,$と$,Q2,$からなる差動増幅器$])$.
$tj1(pnpCE(Q,_),[Q,$のコモンエミッタ$])$.
$tj1(npnEF(Q,_),[Q,$のエミッタフォロワ$])$.
$tj1(opAmp(sdAmp(ecup(Q1,Q2),\_,\_),\_,\_),$
$\quad[Q1,$と$,Q2,$を含むオペレーショナルアンプ$])$.

ここでは回路図や解析木が図形として与えられ，新たな指示対象が視覚的にも認識でき，特定の概念に同定されている状況を想定している．それで，普通名詞 (機能ブロック名) を固有名詞のように用いて対象を指示したが，新たな対象がまだ特定の概念に同定されていない場合は適切でない．構造解析で得られた新たな対象を概念に結びつけるために普通名詞で指示を行うと「そのアクティブロードはアクティブロードである」と言ったおかしな文を生成してしまう．まだ概念に同定されていない初出の対象は複合項をそのまま固有名詞として用いて指示した方が良い．

6·2 日本語文への変換

得られた意味情報 (22) は多数の複合項のリストである．各複合項ごと，外側の関数は述語となるようにテンプレート pj，内側の関数は名詞句となるようにテンプレート tj を適用して日本語に変換する．内側に来ても名詞句テンプレート tj を持たない関数は埋め込み文となるもので，述語テンプレート pj 中の格助詞「は」を「が」に替え，形式名詞「こと」を付加して名詞句を作る．(例)「差動増幅器はオペレーショナルアンプが差分入力を増幅することを可能にする.」

リスト (22) から生成した文の先頭から 30 番目までを示す．回路上の対象の指示を簡潔に行うため，曖昧さの生じないものは機能ブロック名をそのまま用いたのであるが,「エミッタフォロワの入力インピーダンスは高い.」のように，一般概念の説明と区別のできないような文も生成されている．対象の指示と一般概念の指示とを区別するため,「そのエミッタフォロワ」のように「その」を付けるか，やや冗長になるがテンプレート $tj1$ を用いて,「q6 のエミッタフォロワ」のように固有名詞とともに名詞句を作り，対象の指示であることを明確にすることも考えられる．

・オペレーショナルアンプは節点 3，4 間の電圧が入力される．
・オペレーショナルアンプは節点 9，1 間の電圧を出力する．
・節点 3，4 間の電圧は節点 9，1 間の電圧の原因となる．
・差動増幅器はオペレーショナルアンプが差分入力を増幅することを可能にする．
・コモンエミッタはオペレーショナルアンプの電圧利得が高いことを可能にする．
・エミッタフォロワはオペレーショナルアンプの出力インピーダンスが低いことを可能にする．
・エミッタフォロワは節点 8，1 間の電圧が入力される．
・エミッタフォロワは節点 9，1 間の電圧を出力する．
・節点 8，1 間の電圧は節点 9，1 間の電圧の原因となる．
・エミッタフォロワの入力インピーダンスは高い．
・エミッタフォロワの出力インピーダンスは低い．
・エミッタフォロワの電圧利得は 1 に等しい．
・節点 9，1 間の電圧は節点 9 から r2 への電流の原因となる．
・節点 9 から r2 への電流は節点 9，1 間の電圧の原因となる．
・q6 は能動状態にある．
・節点 2，9 間の電圧はコレクタ飽和電圧より大きい．
・節点 8，9 間の電圧はベースエミッタ間順電圧に等しい．
・節点 8 から q6 への電流は 0 より大きい．
・節点 2 から q6 への電流は 0 より大きい．
・節点 8，9 間の電圧は節点 8 から q6 への電流の原因となる．
・節点 8 から q6 への電流は節点 2 から q6 への電流の原因となる．
・コモンエミッタの負荷インピーダンスは高い．
・コモンエミッタは q9 から節点 6 への電流が入力される．
・コモンエミッタは節点 8，1 間の電圧を出力する．
・q9 から節点 6 への電流は節点 8，1 間の電圧の原因となる．
・コモンエミッタの入力インピーダンスは低い．
・コモンエミッタの出力インピーダンスは高い．
・コモンエミッタの電圧利得は高い．
・電流シンクは節点 8 から q5 への電流を制御する．
・q5 は能動状態にある．

7. お わ り に

確定節文法 DCSG を拡張し意味項を導入することで，言語の表層に現われない意味情報と文法構造の関係を直接，規則として定義できるようになった．また，文法規則における終端記号と非終端記号の表記上の区別をなくし，語の意味と文法構造から生じる意味を統一的に扱えるようにした．すなわち，拡張 DCSG は言語の文法構造と意味を同時に特徴づけることができる．

拡張 DCSG は構造を中心に意味情報を付加するので，語彙の豊富な自然言語よりも，人工物の構造とその機能の定義に有効に働く．回路の動作・機能を回路構造から生じる意味としてとらえ，回路知識を拡張 DCSG の規則として定義することができた．この規則を用いて回路の構造解析を行うと，回路の動作・機能が意味情報として導かれた．

導かれた意味情報を読み易くするために，テンプレートを用いて日本語文へ変換した．簡潔な名詞句による対象の指示や述語・名詞句変換により，一応，日本語として読める文が得られた．しかし，得られた文は回路の説明という観点からは不十分である．説明にはユーザが何を知りたいか，どのように内容を順序立てて述べるか，何を述べ何を省略するか，また，文脈を考慮した適切な代

名詞の使用なども考えることが必要になる [田中穂積 99 (3,4 章)].

SPICE[Tuinenga 88] のような回路シミュレータは与えられた回路の電圧・電流を定量的に求めることができるので，回路の試作と計測作業をコンピュータで置き替えることができる．しかし，回路シミュレータはブラックボックスとして働くので，ユーザが出力結果から回路の動作原理を直接的に理解するようなことはない．一方，本システムは与えられた回路を言語として読み，定性的な動作・機能を導出する．システムは回路理解のシミュレータとして働き，技術者の回路理解を助けるコンサルタントの役割を果たす．二つのシステムは相補的なものなので，連携して用いるのがよい．そのためには回路の表記法やユーザインタフェースなどを統一することも重要になろう．

言語から語順の制約を取り除いたことは階層的に構成される多くの人工物を言語的な方法で取り扱うを可能にしている．そこで，DCSG を広く人工物の知識表現に用いる提案がなされている [Tanaka 96]．拡張 DCSG ではさらに構造と機能の関係も直接的に定義できるので，より強力な知識表現言語として働くことになる．

論理プログラムは入出力の関係を自由にとれるので，回路の機能を入力して回路の構造を出力する設計問題も考えることができる．しかし，これまで定義した規則は解析を中心に考えて機能と電気的な動作を一緒に扱ったので，そのまま設計に用いるにはやや無理がある．次の段階として，機能や仕様とそれを実現する構造や条件の観点から，新たに設計・生成型の規則を定義するのがよいと考えている．

設計・生成型の簡単な応用例として「パソコン組み立て部品選択システム」を考えると，文法規則はおよそ次ぎのようになる．ケースとマザーボード，マザーボードと CPU，OS と各種カードなど，依存関係のあるものは意味項を用いて情報を受渡し，選択制限を加えればよい．「サウンド機能が使え，ビデオキャプチャー可能な Linux マシンを作りたい」などの機能に対する要求から，組み立てに必要な部品の集合を無語順言語の文として導くことができよう．

パソコン $\longrightarrow$ OS，本体，ディスプレー，
　　キーボード，マウス．
$OS \longrightarrow Linux;\ BSD;\ WinMe;\ Win2k.$
本体 $\longrightarrow$ ケース，マザーボード，CPU，
　　グラフィックカード，LAN カード，
　　ハードディスク，CD ドライブ，
　　FD ドライブ．

謝　　辞

このシステムは Minerva，Bin-Prolog，GNU-Prolog を用いて開発を行った．DCSG-Converter の開発にあたり IF-Computer 社の Oskar Bartenstein 博士に御協力頂いた．GNU-Prolog における漢字の扱いは東工大の櫻井成一朗先生に教えて頂いた．この研究は文部省の科学研究費補助金 (基盤 C 課題 08680418) を得て行った．

◇ 参　考　文　献 ◇

[IC 76] *101 Analog IC Designs*, Interdesign Inc., Sunnyvale, CA (1976).

[Matsumoto 83] Matsumoto, Y.,et al.: A Bottom-Up Parser Embedded in Prolog, *New Generation Computing*, Vol.1, No.2, pp.145-158 (1983).

[西田 81] 西田富士夫: 言語情報処理, **コロナ社** (1981).

[Pereira 80] Pereira, F. and Warren, D.: Definite Clause Grammars for Language Analysis, *Artificial Intell.*, Vol. 13, pp. 231–278 (1980).

[田中穂積 99] 田中穂積 (監修): 自然言語処理 –基礎と応用–, **電子情報通信学会** (1999).

[田中卓史 87] 田中卓史: 集合型言語の確定節文法 DCSG と応用, **情報処理学会論文誌**, Vol.28, No.10, pp. 1013–1020 (1987).

[Tanaka 91] Tanaka, T.: Definite Clause Set Grammars: A Formalism for Problem Solving, *J. Logic Programming*, Vol. 10, No. 1, pp. 1–17 (1991).

[Tanaka 93] Tanaka, T.: Parsing Circuit Topology in A Logic Grammar, *IEEE-Trans. Knowledge and Data Eng.*, Vol. 5, No. 2, pp. 225–239 (1993).

[Tanaka 96] Tanaka, T.: DCSG: A Logic Grammar for Knowledge Representation of Hierarchically Organized Objects, *in Tanaka,T., Ohsuga,S., Ali,M.(eds.) Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, Gordon and Breach Pub., pp.35–42 (1996).

[Tanaka 99] Tanaka, T. and Bartenstein, O.: DCSG-Converters in Yacc/Lex and Prolog, *Proc. 12th International Conference on Applications of Prolog*, pp. 44–49 (1999).

[Tuinenga 88] Tuinenga, P.W.: *SPICE – A Guide to Circuit Simulation & Analysis Using PSpice*, Prentice-Hall (1988).

〔担当委員：外山勝彦〕

2000 年 12 月 18 日　受理

著　者　紹　介

田中　卓史（正会員）

1967 年九州大学工学部電子工学科卒業．1972 年同大学院工学研究科博士課程了，同大学助手，1977 年国立国語研究所研究員，主任研究官を経て 1988 年より福岡工業大学教授 (情報工学科)，1982-83 年 Yale 大学計算機科学科ポストドクトラルフェロー，1995-96 年 ISAI 副会長，工学博士，電子情報通信学会，情報処理学会，認知科学会，AAAI，IEEE-CS，ISAI，AIUEO 各会員．