グラフィック・エディタ用図形定義文法の設計

3K-8

伊藤 篤、宇都宮 栄二、角田良明、若原 恭

国際電信電話株式会社 上福岡研究所

1．まえがき

近年、情報化社会の発展に伴い通信ソフトウェアの開発・保守に対する要求は急激に増加し続けている。このため、通信ソフトウェア開発・保守の基本である要求仕様をより効率良く開発・保守できる作業環境が求められている。

我々は、ＣＣＩＴＴ標準の仕様記述言語ＳＤＬ([1])をベースにして、通信ソフトウェア要求仕様の作成を支援する要求仕様定義支援システムＥＳＣＯＲＴ(Environment for Specifying Communications Software Requirements)の研究開発を進めており、これまでに要求仕様の検証、プロトタイピング、最適化などを支援するツール([2])、要求仕様の記述・編集を効率良く行なうために有用な構文エディタ機能、および検証・最適化等の諸機能を有機的に統合した環境としての要求仕様定義支援用グラフィック・エディタの検討・作成を行なってきた。([3], [4])

このような、要求仕様定義支援用グラフィック・エディタでは、表示すべき図形やそれらの関係は、対象とする仕様記述言語やツールの種類およびそれらに対する修正により変化する。そのため、仕様記述言語やツールの違いに応じて自由にグラフィック・エディタの仕様を記述し、それに基づいて構文指向グラフィック・エディタを生成できる機能が必要である。

現在、上記機能を実現するため、属性文法を利用したグラフィック・エディタ・ジェネレータについて検討を行なっている。([5])

本稿では、属性文法を利用したグラフィック・エディタ・ジェネレータを記述する属性図形定義文法の基礎となるグラフィック・エディタの仕様を形式的に、かつ理解しやすく記述することのできる図形定義文法に必要な機能について問題点を明らかにし、その問題点を解決すべく設計したグラフィック・エディタ用図形定義文法を提案する。

2．図形定義文法の問題点

エディタの仕様変更や、複数の言語仕様に柔軟に対応して構文エディタを生成できるエディタ・ジェネレータについてはこれまでにいろいろな研究がなされている。しかし、大部分はC言語やPascalのような一次元言語を対象としたものであり([6])、SDL/GRやPADのような図形に基づいた二次元言語を対象とするものではなかった。

その理由としてはいろいろ考えられるが、ひとことで言えば、二次元言語は、図形の位置関係によって意味を表すことが重要な機能であるにもかかわらず図形間の関係を表現する一般的な方法が確立されていなかったということである。このため満足のいく図形エディタを作成することができなかった。

以下に、図形エディタ・ジェネレータを作成するうえでもっとも重要である図形間の関係の表現方法について比較する。

単純な方法として、一次元言語仕様の記述法をそのまま二次元言語に適用することが考えられる。しかし、図形は、二次元的な広がりを持っているため図形間の関係を一次元言語と同様にBNFを用いて前後関係と階層関係からのみ記述することは困難である。

二番目の方法としてグラフ文法を二次元言語の記述に利用する方法がある。([7])しかし、グラフ文法を利用した場合、図形間のつながり方は記述可能であるが、そのつながりの持つ意味をエディタの仕様上に表現することはできない。また、ある図形の中に別の図形が包含されるような図形関係を表現することは困難である。

三番目の方法として、SDL/GRでとられているような図形の関係を表現するメタシンボルを用いる方法がある。SDL/GRでは、BNF記法を拡張し set, is connected to, contains, is followed by, ia associated withという五種類のメタシンボルを追加することにより図形間の関係を記述している。この方法は、図形の関係を陽に記述できるという点から上記の二つの方法よりも優れている。しかし、これらのメタシンボルは、図形間の関係を記述するためには不足していたり、その意味があいまいな部分があるためエディタの仕様の記述には不十分である。また、SDL/GRのメタシンボルはBNF記法の一部であるため実際にエディタ上で作成・表示された図形の内部表現上には現われてこないため生成された図形をメタシンボルで表現される関係を有効に利用して図形の挿入、削除等の操作を行なうことができないという欠点がある。

3．グラフィック・エディタ用図形定義文法

以上の三種類の方法を比較した結果、図形間の関係の意味をわかりやすく記述できることがもっとも重要であるという観点から、メタシンボルを用いた図式記述言語がもっとも優れた方法であると考えられる。そこで、本稿では、メタシンボルを利用した図形エディタ・ジェネレータの入力となる図形言語仕様の記述に適した図形定義文法を提案する。

以下に、SDL/GRのメタシンボルについての問題点を整理し、それらを解決する方法を示し、さらに図形言語仕様の記述に適したメタシンボルのありかたについて述べる。

すでに述べたように、SDL/GRでは、メタシンボルはBNF記法の一部として扱われているため言語仕様の記述に利用されるに留まっており、その機能を十分に有効活用しているとはいえない。従って、メタシンボルにより与えられる情報を有効利用するため、メタシンボルを図形と図形の間に置かれる関係演算子として図形表現の中に取り込むことにより、図形間の関係を図形操作の中でも十分に利用できるようにする。これ以降、この関係演算子をBNFのメタシンボルと区別するためにリレーションと呼ぶこととする。

また、リレーションは図形間の関係の意味を十分に表現できるものでなければならないが、SDL/GRで利用しているメタシンボル（リレーション）にはいくつかの問題点があるため、次のような変更を加える必要がある。

A Design of Graphics Grammar

Atsushi ITO, Eiji UTSUNOMIYA, Yoshiaki KAKUDA, Yasushi WAKAHARA

KDD Kamifukuoka R & D Laboratories

・setは後置演算子であり、その左オペランドの範囲を中括弧で示すため、多数の中括弧がある場合、setに対応する中括弧を判別することが困難である。そこで、setを前置演算子に変更するとともに、その右オペランドの終わりをendsetで示すこととする。また、その他のリレーションについてもオペランドを中括弧で示す必要のあるものは同様の方法でオペランドの終わりを示すこととする。

・is connected toの意味として、ある図形が二つの図形の間にあることを示す場合と、ある図形が別の図形に接続していることを示す場合の二種類があるためその意味が不明確になっている。そこで、ある図形が二つの図形の間にあることを示す場合にis connected toを用い、ある図形が別の図形に接続していることを示す場合はjoins toという新しいリレーションを導入することによりこの二種類の図形関係を区別する。例を図１に示す。

・is followed byはある図形に続いて別の図形が接続しているという接続関係を示すと同時に、それらの間にフローラインが生成されることを示す。このように本来図形と図形の関係のみを示すメタシンボルが図形の生成まで行なうことはリレーションの本来の意味を逸脱している。そこで、is followed byの意味をフローラインの生成を行なわないように変更する。もし、フローラインが必要であればプロダクションルールの中にフローラインの生成を記述することとする。

さらに、シーケンス図、ツリー図、SOD(state overview diagram)等のSDL/GR以外の表現形式や、エラー表示にも対応できることが必要である。特にこれらの表示のうちエラー表示はすでに存在している図形の上にエラーメッセージを表示したり、エラーの発生した場所を理解しやすいように色が付けられたり点線で囲まれた図形により表示しなければならない。このためには、すでにある図形の上に別の図形を重ねた表示することができなければならない。そこで、ある図形がすでに存在している図形の上におかれることを示すis overという図形関係をあらたに導入することとする。例を図２に示す。

以上の議論を踏まえて、グラフィック・エディタ用図形定義文法Ｇを次のような５項組として定義する。

Ｇ＝（N,T,R,P,Z）；ただし、N：非終端記号の集合、T：終端記号の集合、R：図形の関係を表すリレーションの集合（R∈T）、P：生成規則の集合、Z：開始記号（Z∈N）

ここで、リレーションは、set, is associated with, is connected to, contains, is followed by, joins to, is overの七種類とする。これらのリレーションの意味を表１に示す。

４．あとがき

本稿では、現在作成中の要求仕様定義支援システムＥＳＣＯＲＴの中核として要求仕様の記述・編集、ならびに検証結果・最適化結果等の機能を有機的に統合するグラフィック・エディタを生成することを目的とし、グラフィック・エディタの仕様を形式的かつ理解しやすく記述することのできる文法に必要な機能について問題点を明らかにするとともに、それら問題点を解決するグラフィック・エディタ用図形定義文法を提案した。

今後は、リレーションの種類および意味についてさらに検討すると共に、属性文法を応用した実用的なグラフィック・エディタ・ジェネレータの構成について検討・試作し、このグラフィック・エディタ用図形定義文法の有効性を確認する予定である。

謝辞　日頃ご指導いただくＫＤＤ上福岡研究所小野所長、湯口次長、安藤主幹研究員、浦野主幹研究員に感謝する。また、本研究を進めるにあたって有益なご助言とご意見をいただいた通信ソフトウェア研究室小西室長に感謝する。

文献

[1]CCITT Recommendation Z.100　1988年版
[2]若原、角田、伊藤：”通信ソフトウェア要求仕様作成支援システムESCORT一検証サブシステムVAVES一”、ソフトウェアシンポジウム(1988.6)
[3]八木、伊藤、角田、若原：”図式言語向きエディタジェネレータにおける図式言語規定法の設計”、第35回情処全大 7k-7 (1987.9)
[4]伊藤、角田、若原:”通信ソフトウェア要求仕様作成支援システムESCORTにおけるデータ構造”、昭和63年電通春期全大 B-347 (1988.3)
[5]伊藤、角田、若原:”属性文法の図式仕様処理への応用”、第36回情処全大 2L-2 (1988.3)
[6]Reps,T & Teitelbaum,T:"The Synthesizer Generator",Proc.ACM Soft. Eng. Symp. on PracticalSoft. Dev. Env.,SIGPLAN Notices,Vol.19,No.5,pp.42-48 (1984)
[7]西野:”属性グラフ文法とそのHichart型プログラム図式に対するエディタへの応用”、コンピュータソフトウェア、Vol.5, No.2,pp81-92 (1988.4)

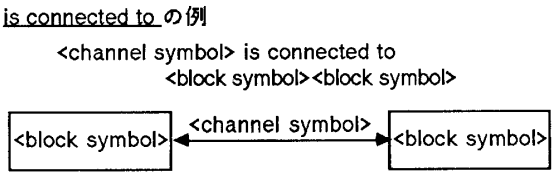

図１．is connected to と joins to の例

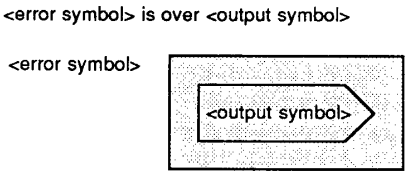

図２．is over の例

* set	set <A> … <Z> endset <A>… <Z>の間に順序がない
# is connected to	<A> is connected to <B>,<C> <A>が<B>,<C>に接続する
@ joins to	<A> joins to <B> <A>に<B>が接続する
# is followed by	<A> is followed by <B> <A>に<B>が続く
* is associated with	<A> is associated with <B>… endassociate <A>の近傍に<B>… がある
* contains	<A> contains <B> … endcontain <A>に<B>… が含まれる
@ is over	<A> is over <B> <A>が<B>の上にある

@　あらたに追加したもの　　#　意味を変更したもの
*　構文を変更したもの

表１．リレーションの意味