4E−1　属性文法に基づくグラフィカルユーザインタフェース生成系とその評価

金子 正俊　林 謙一†　吉羽 幹夫‡　脇田 建　佐々 政孝
東京工業大学 理学部 情報科学科　†(旧)筑波大学 理工学研究科　(現)ソニー　‡筑波大学 理工学研究科

1　はじめに

近年、グラフィカルユーザインタフェース(GUI)の有効性、重要性が認識されるようになり、GUIは急速に広まりつつあるが、その開発にはかなりの労力を要することが問題とされている。そこで本研究室ではGUI生成系wingを開発している[3]。wingは属性文法風のGUI仕様記述とC言語で書かれた補助関数から、既存の文字端末用アプリケーションに付加するGUIを生成する。

wingには次に挙げる2つの特徴がある。(1) GUI仕様の記述方法に属性文法を採用した。(2) GUIを付加する対象のアプリケーションと独立したプログラムを生成する。われわれはwingの評価のため、同じ仕様のGUIのC言語のみによる記述とwingによる記述を比較した。その結果、記述量が大幅に減ることを確認した。

2　wing

wingを設計した当初の目的は、既存の文字端末アプリケーションにGUIを与えることであった。このためには文字端末アプリケーションのソースを書き直してGUIの機能を付加する方法と、アプリケーションとは独立したGUIを提供するプログラムと文字端末アプリケーションを組み合わせる方法[1]が考えられる。普通はアプリケーションのソースを手に入れることは困難であり、手に入れられたとしても実装を理解することが困難なので、われわれは後者の立場に立つことにした。

図1はwingを用いて文字端末アプリケーションⒶにGUIを与える方法を示している。プログラマはGUIの形式的な記述Ⓑを書き、wingはそれをGUIのソースプログラム群Ⓒに自動変換する。これらのプログラム群はCコンパイラによりアプリケーションのGUI部分Ⓓにコンパイルされる。端末アプリケーションとこのGUIを並列実行することによって、GUIつきのアプリケーションが実行できるⒺ。この設計においては、(1) 文字端末アプリケーションとGUIの独立性、(2) GUIの仕様の記述性、(3) 文字端末アプリケーションとGUIの通信の記述 が重要である。

独立性　wingを使ってアプリケーションにGUIを付加する時に、GUIはアプリケーションとは独立なプログラムとして作成される。われわれのアプローチではGUI部分(GUIフロントエンド)とアプリケーションが共有データを持たないために、実行時にはアプリケーションとGUIが互いに通信することによりユーザの入力やアプリケーションの出力結果などを交換しあう。このために、オペレーティングシステムのインタープロセスコミュニケーションの機能を用いて、GUIの出力をアプリケーションの標準入力に、GUIの入力をアプリケーションの標準出力につなぐことで情報をやりとりする。実際には、wingではGUIをktermを介してアプリケーションにつなげている。この方法には、アプリケーションのソースコードを書き変える必要がない、文字端末ウィンドウでそのアプリケーションが今まで通り実行できる、などの利点がある。

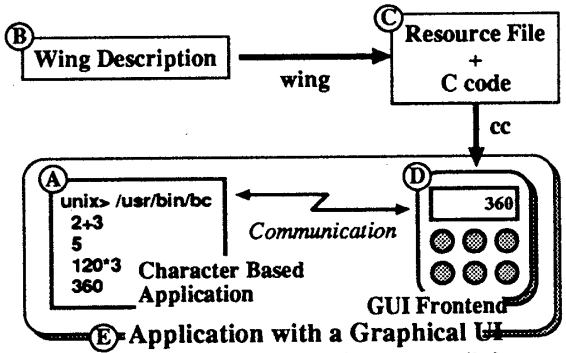

図1: GUI生成系wingを利用したGUIの作成

図2: ボタンを2つ持つメニューの例

記述性　GUI仕様記述の記述性、可読性を向上させるために、wingのGUI仕様記述法は属性文法に基づいて設計された。定式化としてパラダイムの確立した属性文法に基づくことで、概念を把握しにくいGUI記述言語に一貫した枠組を与えることができた。また、widgetの階層構造の概念は属性文法の概念と馴染みやすく、その点からも属性文法の枠組は有効である。ここでいう階層構造とは、それぞれのwidgetが自分が様々なリソースを継承すべき親widgetを持つことを指している。wingでは、widgetの階層構造を属性文法の構文規則に、widgetのリソースの決め方やwidgetの振舞いを属性文法の意味規則に対応させる。wing記述においては、子どもを持つwidgetは非終端記号、持たないものは終端記号とする。

以下にwingを用いて図2に示したボタンを2つ持ったGUIを記述した例を示す。

```
%nonterm   menu:    SimpleMenuWidget;     ←非終端記号
%terminal  menubut: MenuButtonWidget;     ←終端記号
           item:    SmeBSBObject;         ←終端記号
--- 略 ---
menubut { %dynamic %popup menu; };  (a)
menu : item[1] item[2]                    ←構文規則
    { %inherit font;              (c) ←意味規則
      item[1].label = "item1";        (b)
      item[2].label = "item2"; };
```

この例の前半で定義しているmenu, menubut, itemは文法記号とそのwidgetのクラスを表す。menuはポップアップメニューの、menubutはmenuをポップアップさせるボタンの、itemはmenuの項目のwidgetである。後半では、属性文法に基づいてwidgetの階層構造と、{}内にリソースの値を記述している。この例の構文規則は、menuはitem[1]とitem[2]の2つの子どもを持つことを表している。menu, menubutの様に構文規則に文法記号を書くと、それがwidgetのインスタンスとして扱われる。またitem[1]の様に文法記号名に任意の文字列のラベルをつけることで、その文法記号の複数のインスタンスを表すことが出来る。

GUIには静的な要素と動的な要素がある。GUIの静的な要素は、widgetの階層構造と配置、widgetの生成時に設定されるリ

Implementation and Evaluation of a GUI Generation System Based on a Attribute Grammar
M. Kaneko, K. Hayashi, M. Yoshiba, K. Wakita, and M. Sassa
Tokyo Institute of Technology & University of Tsukuba

ソース(フォントやコールバック)などである。動的な要素は、メニューのポップアップなど GUI の変化、コールバック関数の呼び出しなどである。上記の(b)の意味規則は属性の設定をあらわす。この規則は widget 生成時に一度だけ評価されるもので、静的である。これは属性文法の静的意味規則に相当する。一方(a)の意味規則は自分(menubut widget)がマウスでクリックされると menu widget をポップアップさせることを表す。このような規則は生成時にではなく、クリックされるたびに評価されるべきもので、動的である。従って動的意味規則と考える。(a) の %dynamic は、この意味規則が動的であることを表している。

属性の継承は属性文法における概念だが、X Toolkit においても widget のある種のリソースは親 widget から継承される。(c) の %inherit は、menu widget から item[1], item[2] widget への属性(font)の継承を指定する略記法である。wing においては、widget のクラスの元々持っている属性の継承を行なうだけでなく、任意の widget のクラスに必要な属性を付け足して、階層構造の先まで継承を行なうことが出来るようになっている。属性文法による属性の評価式には、C 言語に準じた演算子による演算や、C 言語で記述した補助関数の呼び出し、ローカル属性の使用、条件式が記述できる。

通信　wing の生成する GUI はアプリケーションと独立したものである。従って GUI はアプリケーションの出力に対して適切な応答が出来なければならない。そのためには、GUI がアプリケーションの出力を監視してアプリケーションの状態を判断し、その上で任意の応答をアプリケーションに返すために、われわれは正規表現によるパターンマッチングを用いる。wing 記述には前出の意味規則記述部の他にパターン記述部があり、以下のように正規表現と意味規則を記述する。

```
%%PATTERN "\(\-?[0-9][0-9]*\)"  { disp.label=$1; };
```

アプリケーションからの出力が左側の文字列にマッチすると、右側の意味規則が評価される。従ってこれは動的である。この例では計算プログラムの出力した数値を disp という widget の label リソースに設定している。パターンは複数記述できるが、実行時に局面によってどのパターンを有効にするか、といったことは記述しづらい。さらに、複数の文字端末アプリケーションを対象とした GUI を記述する場合もある。そこで、任意の文字列をアプリケーションに出力したり、アプリケーションからの入力にパターンマッチングしたりする関数が用意されている。これらを用いて、意味規則記述部でも任意にアプリケーションと通信が行なえる。

3 実装

図1に示したように、属性文法に基づいた記述の部分は wing の処理系により C 言語で記述されたプログラムに変換される。その際ユーザが(必要に応じて)記述した補助関数とリンクされる。

wing は、wing 記述から GUI フロントエンドと共にリソースファイルを生成し(図1)、widget の生成に利用する。widget の生成は C の関数で一つづつ行なうのではなく、Wcl(Widget Creation Library)[2] の関数を利用して行なわれる。Wcl により、専用のリソースファイル内に記述された widget の階層構造に従って widget をまとめて生成することが出来る。

4 評価

4.1 記述量

wing による GUI 記述を評価するために、3つの例で比較を行なった。wing による記述と比較するのは、C 言語と X Toolkit のみを使用して記述されたプログラムである。

表1: wing による記述と C 言語 +X Toolkit による記述の比較

	wing	C 言語 + Xt	wing/C
bc	188 (432;3939)	247 (619;6157)	0.76
pl0dbg	103 (236;2102)	226 (562;5269)	0.45
Xrj	1288 (2572;32906)	1669 (3277;36672)	0.77

表1の数字は、行数(語数;文字数)である。bc とは UNIX の計算プログラムである。pl0dbg とは本研究室で作成された PL/0 言語用デバッガ [4] である。この2つは元々文字端末用アプリケーションであるが、これに2通りの方法で GUI フロントエンドを付加した。Xrj[5] は本研究室で作成された統合プログラミング環境で、当初は C 言語 + X Toolkit のみで作成されたが、wing 上に移植された。

記述量については行数、語数、文字数の全ての項目において wing による記述の方が記述量が少なくなっている。この差の大きな原因は、C 言語で記述する場合、アプリケーションとの通信機構を記述する必要があることである。wing で記述する場合には、wing によって GUI フロントエンドに通信機能が組み込まれるのでこの部分が必要ない。

4.2 記述性と可読性

2節の例からわかるように、widget の配置関係や親子関係、リソースの設定が視認でき、わかりやすくて変更も容易である。一つの widget に関する設定が局所化しているために、ある変更の影響が及ぶ範囲が楽に特定できる。これにより可読性、保守性が C 言語のみの記述より格段に向上したと考えられる。

しかし記述性については、以下のような制限がある。(1) Athena widget のみしか使えないことなどの実現上の制限がある。(2) 動的に widget が生成、消滅するような GUI は記述できない。(1) の制限は移植性を考慮したためである。(2) の制限は属性文法では属性評価中の解析木の動的な変化を考慮していないために発生した。メニューのポップアップはあらかじめ生成された widget が画面に表示されるだけなので、動的な生成には当たらず、可能である。

5 問題点と今後の課題

wing の問題点として、ユーザが自由にクラスを定義できないこと、そのため記述に冗長性があること、などがある。現在のところ wing におけるクラスとは X Toolkit のクラスのことである。したがって同じような機能を持つ widget をユーザが統一的に扱うことが出来ず、同じような機能の widget を使う時は同じような記述が連なることになる。

前節でも述べたが、どのようなアプリケーションに対する GUI でも wing で記述できるわけではない。それは基本的に属性文法という枠組を仮定しているからである。生成される GUI の widget のうち構文規則に現れないもの、すなわちユーザが明示的に宣言したもの以外の widget、へのアクセス手段がないなどの制限がある。これらについては今後の課題である。

参考文献

[1] J. Rudolf and C. Waite. Completing the Job of Interface Design. *IEEE Software*, pp. 11 – 22, Nov. 1992.

[2] D. E. Smyth. *Wcl - Widget Creation Library.* Jet Propulsion Labs, Jan. 1991.

[3] 林謙一. 属性文法に基づくグラフィカルユーザインタフェースの記述と生成. 筑波大学理工学研究科 修士論文, 1992.

[4] 田淵聡. 木属性文法によるデバッガの生成. 筑波大学第三学群情報学類 卒業論文, 1991.

[5] 吉羽幹夫. GUI 生成系 wing によるプログラミング環境の実現. 筑波大学第三学群情報学類 卒業論文, 1992.