言語処理系に対する適合性試験プログラムの設計法

1J－1

NTT ソフトウェア研究所
橋本 辰範　宮崎 祐

1 はじめに

コンピュータのオープンシステム化、ダウンサイジングが進むにつれ、ユーザ主導によるマルチベンダシステム構築の検討が活発化してきた。ただし、ユーザ側がマルチベンダシステムの利点を享受するには、アプリケーションプログラムの一元化が一つの大きな条件となる。そのため、ユーザ側から各ベンダに対しアプリケーションプログラムインタフェースの共通化が求められており、複数ベンダ間でのプログラミング言語処理系の機能統一などが進められている。ここで、各ベンダの言語処理系の機能が実際に一致しているかどうかを確認するための手段として、任意の処理系が特定の仕様を満足していることを検査する適合性試験が注目されている。厳密に言うと、適合性試験の目的は、試験対象処理系が仕様における規格合致性や仕様準拠性の定義に基づいた処理系であることを確認することである。一般に、言語処理系の適合性を検査するには、基準となる言語仕様と、検査対象処理系上でその適合性についての判定が可能な試験プログラム集合が必要となる。もちろんこの試験プログラム集合には仕様に対する漏れや誤りがあってはならないし、システム構築の観点から異言語間でもなるべく均質であることが望ましい。ところが、言語仕様の形式的記述はその複雑さゆえ困難であり、形式的記述を基に試験プログラム集合を生成するという既知の手法は適用できない。そこで、言語仕様という限られた対象のもつ特性に着目し、自然言語記述を扱いながら、均質な適合性試験プログラムを設計する方法を考案した。

2 試験プログラム作成手順

適合性試験プログラム集合を作成するためには、一般に自然言語で記述されている言語仕様を可能な限り正確に解釈しなければならない。この仕様解釈における混乱を回避するために、POSIX での適合性試験手法 [1] に準じ系統立てられた手順と、言語仕様記述の分析に基づいた独自の手法とで構成する設計法を定めた。以下に試験プログラム作成手順を述べる。

(1) 仕様記述の分解
まず、全仕様記述を仕様書の章や節を頼りに機能単位に区切る。この区切られた単位を仕様要素と呼び、他の機能から独立した最小の仕様記述部分と考える。次に各仕様要素を句点で区切る。この句点で区切られた個々の文を仕様定義文と呼ぶ。さらに、仕様定義文で定義されている個々の機能や動作をそれぞれ独立した文章記述に分解する。この文章をアサーションと呼び、仕様要素に対する適合性を確認するために必要な試験最小単位とする。

(2) 試験項目の設計
任意の試験対象処理系に対し、個々のアサーションの真偽を判定するための具体的な試験内容を検討し、試験項目として明文化する。

(3) 試験プログラムの作成
試験項目集合を基に具体的試験方法を検討し、試験対象処理系への入力となる試験プログラム集合を作成する。

試験項目は複数のアサーションから、試験プログラムは複数の試験項目から構成してもよい（図1）。

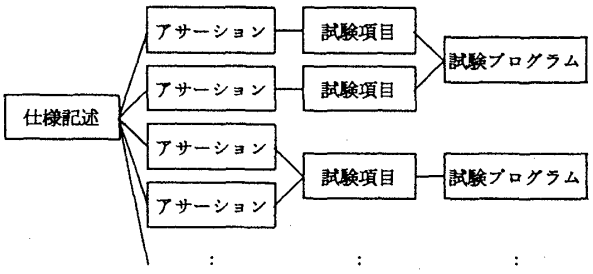

図 1: 仕様記述と試験プログラム

3 言語仕様記述の属性

一般の文章に較べると、言語仕様書の文章は、文脈への依存度が低く、あいまいさも少ない。また、言語仕様を記述するための文章としてある種の共通なスタイルが確立されている。これらの特性から、言語仕様書に記載されている文、あるいは節をある限られた属性値の集合で分類することが可能となる（表1）。そして、この属性値毎にそれぞれ一意の解釈方法を規定することにより、対象とする仕様記述が本来言及している要求条件をより的確、均質に把握することができる。

4 試験プログラムの設計

4.1 仕様定義文への分解

アサーションを容易に得るために仕様記述の物理的な分割を行なう。ここでは、仕様記述の表面的な構造にのみ着目すれば十分であり、属性「記述形式」で分類しそれぞれに適した分解を行なう。また、属性「記述対象」

Designing the Conformance Test Program for Programming Language Processors
Tatsunori HASHIMOTO, Tasuku MIYAZAKI
NTT Software Laboratories

表 1: 言語仕様書に見られる文、節の属性

属性	属性値	内容
記述対象	処理系作成者	処理系の機能や動作を示し処理系を実装する上で必要不可欠な記述
	ユーザ	処理系利用者に対する制約やガイドラインを示した記述
記述形式	自然言語記述	自然言語を用いた文または語による記述
	形式的記述	BNF など厳密に定められた文法規則に従った形式的な記述
規則	構文則	言語の構文に関する規則の記述
	意味則	言語の意味に関する規則の記述
独立性	基幹	対象とする文だけで内容を明確に把握できる記述
	付随	他の文と組み合わせなければその内容が明確とならない記述
一意性	一意	一意の実装を規定している記述
	任意	処理系依存など、任意または選択による実装を許す記述
表現	肯定	肯定文を用いた記述
	否定	否定文を用いた記述

に着目し、言語仕様書に存在するユーザ向けの記述をアサーションの対象から排除する。

4.2 アサーション抽出

仕様定義文は独立した文ではあるものの、その内容を解釈すると、処理系に対する複数の要求条件が記述されていることが一般的である。これらの要求条件を一つ一つの独立した文に分解することをアサーション抽出と呼ぶ。ここで、アサーションとは、真偽の値をもちうる命題であり、単一の要求条件を表明し、単独で内容が理解できる独立した文であると定義する。以下にアサーション抽出のポイントを示す。

(1) 仕様定義文の展開
属性「記述形式」の属性値それぞれの特性に応じ、対象とする文を文脈から解放し、かつその中に混在する命題を分割する。このとき、全アサーションの論理和で処理系全体の適合性が判定できるように分割方法を決定する。（例えば、属性「一意性」の属性値「任意」に触れる記述部分は分割しないことなど。）

(2) 構文則と意味則の暗黙的解釈
属性「規則」の属性値「構文則」「意味則」それぞれに対し、「解釈できる」「動作する」という記述が省略されている可能性があることを意識する。

(3) 処理系定義の扱い
処理系定義を含む記述は決して不要な仕様ではない。この記述も一つの命題として取り扱う。

(4) アサーションの独立
アサーション間の意味的な依存関係をなくすために、属性「独立性」に着目し、依存の背景となる条件記述があればそれをすべて明示的に記述する。

4.3 試験項目作成

試験項目は、あるアサーションの真偽を判定するために、何を確認するのかを明確に記述した文章であると定義する。以下に試験項目作成のポイントを述べる。

(1) 試験プログラムのクラス
そのアサーションを判定する試験が正常、異常、限界試験のいずれにあたるかを意識して項目化する。

(2) 未定義とオーバースペック
オーバースペックをもつ処理系を適合とみなす仕様では、未定義仕様に関する試験は一切行なってはならない。特に、属性「表現」が「否定」である記述については十分な検討が必要である。

(3) アサーションの統合
同一の条件で複数のアサーションが確認できる場合、試験項目においてそれらを統合する。

(4) 翻訳限界、補集合表現
同値関係を考慮し、限界値などの代表値を用いた試験を想定しなければならない場合がある。

4.4 試験プログラム設計

以下に設計のポイントとなる事項を述べる。

(1) 試験項目の組合せ
適合性試験は仕様解釈の正当性の保証を目的としており、製品品質について言及するものではない。よって、試験レベルとしては全要素試験 [1] で十分であり、試験項目の組合せまで考慮した試験プログラムは必須ではない。

(2) 試験項目との対応
試験プログラムの可読性を高めるとともに、ある項目の試験で発生した異常の影響を他の項目の試験に及ぼさないために、一つの試験プログラムは一連の切り離せない試験項目だけで構成する。

(3) 処理系依存
処理系に依存する記述を避ける。避けられない場合は、容易に書換え可能となるようにしておく。

(4) プログラムとして実現できない場合
チェックリスト形式による書類審査など別の試験手段を検討する。

5 おわりに

本設計法により COBOL, C, FORTRAN の独自追加仕様に対する適合性試験プログラムを開発し、均質で漏れがない試験プログラムが作成できることを確認した。

参考文献

[1] IEEE Std.1003.3,1991, "IEEE Standard for Information Technology - Test Methods for Measuring Conformance to POSIX"