私たちの生活や社会の重要な部分には、大規模なシステムがたくさん稼動しています。
社会のインフラを支える大きなシステムは設計も大掛かりとなり、
設計に誤りがないかを調べる「設計検証」も当然複雑なものとなります。
また、生活に密接に結びつくシステムでは、
たったひとつの設計ミスでも大きな損失につながるため、
設計ミスを見落とし無く確実に見つけることも課題となっています。
時相論理を用いた設計検証がご専門の平石裕実先生に、
論理で設計を検証する方法についてお話しいただきました。

ネットワークメディア学科
平石　裕実 教授

設計の正しさを確かめる論理の力

限界があった従来の設計検証

システムの複雑化・大規模化や、金融・インフラなどに求められる無謬性に対して、従来の設計検証では限界が見えてきました。

1993年にインテルが製造したプロセッサでは、あらかじめ割り算の答えを表にして、その組合せを利用して計算の高速化を試みましたが、表に設計ミスがありました。多くのユーザが交換を希望し、4億7500万ドルもの損失となりました。

EUのロケット・アリアン5の爆発事故では、アリアン4から大幅に強化された推進能力が、姿勢制御プログラムの数値の範囲を超えたことが原因でした。史上最も高くついた設計ミスと言われています。

従来の検証方法のひとつ、論理シミュレーションでは、ユーザの様々な使い方を想定してシミュレーションを行います。しかし、インテルのプロセッサの場合、間違えるのは2万7千年に1回の計算だと言われていますし、アリアン5でも、アリアン4までは問題のないプログラムでした。いずれも想定していない計算が現実に行われてしまったのです。

論理シミュレーションによる設計検証

論理シミュレーションによる設計検証の方法を、自動販売機の設計を例に見てみましょう。コインは50円と100円だけとし、100円で商品1つ、100円を超えると商品1つとお釣(50円)を返すよう設計します。

右図がその設計図です。スタート直後はAの状態で、矢印が状態の遷移(移り変わり)を表します。矢印(遷移枝)に書かれている「100円/商品」などは「利用者の行動/自動販売機の出力」です。

利用者の使い方として、たとえば次のような状態遷移を想定すると

A→50円/何もしない→B…問題なし
A→100円/商品→A…問題なし
A→50円/何もしない→B→50円/商品→A…問題なし
A→50円/何もしない→B→100円/商品＋お釣→C…問題なし
A→50円/何もしない→B→100円/商品＋お釣→C→50円/何もしない→D…問題なし

ここまでは設計ミスが無いように見えます。では、50円、100円、50円、100円、50円と入れるとどうなるでしょう?

A→50円/何もしない→B→100円/商品＋お釣→C→50円/何もしない→D→100円/商品＋お釣→D→50円/商品→C

合計350円の投入に対して、商品3つとお釣2回が出てきました。これはシンプルなシステムですが、実際のシステムは比較にならないほど複雑で、何通りシミュレーションすれば十分なのか、正確には分からないのです。

数学的に設計の正しさを証明する記号モデル検査

1990年代前半、システムの状態や遷移を変数として、論理関数を作り、それを満たす解を求める「記号モデル検査」と呼ばれる数学的な検証方法が提案されました。

論理関数とは、論理式で表された関数で、論理式には通常の四則演算とは異なる論理演算(表1)が適用されます。

先ほどの自動販売機を検証してみましょう。変数x、yで状態A、B、C、Dをそれぞれ(x,y)=(0,0)、(0,1)、(1,0)、(1,1)とし、変数zで50円を0、100円を1と表すと、各遷移枝は表2のようになります。このとき、それぞれの集合を表すと

商品を出した直後の状態集合X={A,C,D}

$$X=x+\bar{y}$$

集合Xから出ている50円の遷移枝の集合Y={a,e,g}

$$Y=X\cdot\bar{z}=(x+\bar{y})\cdot\bar{z}$$

商品を出す遷移枝の集合Z={b,c,d,f,g,h}

$$Z=\bar{x}\bar{y}z+\bar{x}y\bar{z}+\bar{x}yz+x\bar{y}z+xy\bar{z}+xyz=y+z$$

となります。各遷移枝にx,y,zの値を代入して、1となる枝は集合に含まれ、0となる枝は含まれません。

正しい設計であれば、集合Y(商品を出した直後に50円入れた)と集合Z(商品を出す)の共通部分Y∩Zは遷移枝を含まず、必ず0になるはずです。

ところが、Y∩Z=1を満たす遷移枝g(1,1,0)が存在し、gに誤りがあると分かります。

$$Y\cap Z$$
$$=Y\cdot Z$$
$$=(x+\bar{y})\cdot\bar{z}\cdot(y+z)$$

g(1,1,0)を代入すると

$$(1+\bar{1})\cdot\bar{0}\cdot(1+0)$$
$$=(1+0)\cdot1\cdot(1+0)$$
$$=1\cdot1\cdot1=1$$

数学の問題なので解が求まれば、そこで検証完了です。さらに、状態や遷移枝を集合として扱うため、従来よりもずっと大規模なシステムも検証可能なのです。

設計に対する安全性と信頼性を高めるだけではなく、より大規模なシステムの設計へと私たちの創造性を広げてくれるもの、それを実現するのが論理の力なのです。

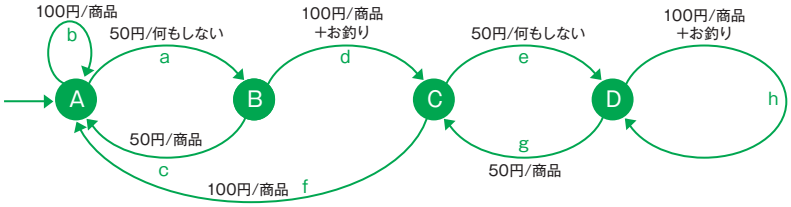

表1　論理演算による和・積・否定

論理和 (or/+)

a	b	a+b
0	0	0
0	1	1
1	0	1
1	1	1

論理積 (and/・)

a	b	a・b
0	0	0
0	1	0
1	0	0
1	1	1

論理否定 (not/$\bar{\ }$)

a	$\bar{a}$
0	1
1	0

表2　遷移枝の符号化

遷移枝	符号 (x,y,z)	表している状態と投入されたコイン
a	0,0,0	状態A、50円
b	0,0,1	状態A、100円
c	0,1,0	状態B、50円
d	0,1,1	状態B、100円
e	1,0,0	状態C、50円
f	1,0,1	状態C、100円
g	1,1,0	状態D、50円
h	1,1,1	状態D、100円