METHODS OF MODEL REDUCTION

© 2011 г. Т.А. Деменкова, Л.В. Твердохлебов*

Moscow State Institute of Radioengineering, Electronics and Automation,
*Politecnico di Milano University
e-mail: demenkova@mirea.ru

Data abstraction is used as a way to reduce the size of the design before the verification. The basic idea is to define a mapping from concrete variables of the model to fewer abstract variables [1]. This mapping consists of a list of concrete variables to be eliminated, and a list of variables with abstract values domain. Such mapping depends on the property under verification, hence the abstraction is repeated each time a property is defined.

In this work we are going to introduce two reduction algorithms based on data abstraction. The first one called Cone of influence (COI) [2], which is variable reduction algorithm where variables removed if they are not affecting the property. This algorithm has been successfully implemented in many tools [3,4]. Making use of the observation that not all the values domain of the remaining variables affect the property, we will introduce a values abstraction algorithm proposed first in [5], where the values domain of the variables are partitioned into two categories depending on the property. Values affecting the property are kept while the others are abstracted.

The basic idea of Cone of Influence reduction is to construct a dependency graph of the program variables, rooted at the variables in the specification. The set of variables in the graph is called the COI of the specification. The variables not included in the COI set cannot influence the validity of the specification and can therefore be removed from the model.

Let’s consider the example in Fig. 1. We want to verify a property about variable a. If we examine its DDG in Fig. 2, we find that a only depends on c which depends on f. Other variables are not related to neither a nor c and f. Removing these variables will not affect the property under verification. We say that the cone of influence of a includes only the variables c and f.

In COI reduction, the model is reduced by removing behaviors when going from the concrete to the abstract system. Since no new behaviors are added by non-deterministic statements, as in the case of over approximation techniques, strong preservation of the properties is achieved. Suppose the properties over the system models are specified by the computational tree logic CTL.

The cone of influence set for a certain variable ν (COIν) is built by fixpoint iteration using forward reachability analysis on the control flow graph; each iteration adds new variables (related to the ones added before) to the list. The iteration stops when there are no more variables to be added. After the COI set is created, only edges with variables included in the set are kept, other edges are removed and a new reduced control flow graph is created. If we apply the cone of influence algorithm for the Verilog program in Example 1, and the property to be verified is about the variable pc, then the cone of influence of this variable is COIpc = {x,pc}. The reduced data dependency graph (DDG) and control flow graph (CFG) of this program are shown in Fig. 2, 3 respectively. Only part of the value domain of the variables usually affects the verification process. For example, if we have a property to check whether a variable will reach a certain value, such as: AF(A > 10 ⇒ B = 1). This property states that eventually on all paths, B will be set if A is greater than 10.
We can understand from this statement that the model checker, in order to verify this property, will look for states where the values of $A$ are greater than 10 and check that $B$ will have the value one. States with $A \leq 10$ are, however, of no interest. If they are abstracted, the property will not be affected and the state space to explore will be smaller, thus avoiding a possible state space explosion problem.

Fig. 1. COI example.

```
Module coi_exp (c,r,a,l)
  input c,r,a;
  output a;
  reg a,b,e,f;
  initial
  begin
    a = 0;
    b = 0;
    e = 1;
    f = 1;
    end;
  always
    begin
      if(c == f)
        a = 1;
      else
        if(d == e)
          b = 1;
        f = f+1;
      end
  endmodule
```

Fig. 2. Simple Verilog program.

```
module exp_1(input ir,input out,input [0:2] x,y,input[0:2] pc);
begin
  if(x<100)
    begin
      pc = 0; y=y+1;
    end
  else pc=1;
  Initial
  begin
    x = x+1;
    end
  'd01: begin
    pc <= 0;
    x <= 0;
    y <= 0;
  endcase
  end
  always
  begin
    case(pc)
      'd02: begin
        if (y == 100) out = ir;
      endcase
    end
endmodule
```

Fig. 2. Simple Verilog program.
In the following, we will introduce the definitions of the key notions used for the values abstraction algorithm.

A test node is a node with one of its outgoing edges is an assignment edge and is labeled by a variable under verification, i.e. included in the property logical expressions. If this latter assignment satisfies the truth of the logical expressions then the test node is a key node $\phi$.

An active path $\tau$ is a CFG path $\pi$ which begins by node $\omega$ and ends with a key node. As more than one path can ends with the same key node $\phi$, the set of active paths having $\phi$ as the end node is $\Pi_{\phi} = \{\tau_1, \ldots, \tau_n\}$.

Let $\pi$ be a finite path in the CFG and $b \in \text{dom}(v)$ a value, we say that a value $b$ is active iff $RC_\pi(v)$ is true or $ST_\pi(v) = b$. Similarly, we say that a value $h$ is deactive iff $RC_\pi(v)$ is false or $ST_\pi(v) \neq b$. $RC_\tau(v)$ and $ST_\tau(v)$ are the Reachability Condition and the State Transformation of the path $T$, respectively.

Intuitively, the active domain, $\text{ACTIVE}(v)$, will contain the values affecting directly the property and will remain unabstracted, while the deactive domain, $\text{DEACTIVE}(v)$, can be abstracted by using a representative value and therefore will contain one single value [6].

- If $\text{ACTIVE}(v) = \emptyset$, then $v$ is a variable not affecting the target statement and $\text{DEACTIVE}(v) = \{d\}$, where $d \in \text{dom}(v)$ and $d$ is the abstract value.
- If $\text{ACTIVE}(v) \neq \emptyset$ and $\text{DEACTIVE}(v) \neq \emptyset$, then $v$ has values affecting the target statement, and these values will not be abstracted. They will be stored in $\text{ACTIVE}(v)$, while the rest of the values will be abstracted.
- If $\text{DEACTIVE}(v) \neq \emptyset$, then the whole domain of $v$ will affect the target assignment.

We know that the values 2 and 100 are active value of the variable pc and y, respectively. By reachability analysis on the path sequence we find that it will not be abstracted as it is not assigned anywhere. The values domain of pc are 0,1,2 with $\text{ACTIVE}(pc) = \{2\}$, $\text{DEACTIVE}(pc) = \{1\}$. The abstract values domains of the variable y and its dependency variable x are be partitioned respectively as $\text{ACTIVE}(y) = \{100\}$, $\text{DEACTIVE}(y) = \{0\}$, and $\text{ACTIVE}(x) = \{\}$.
and $\text{DEACTIVE (x) = \{0\}}$. For the other variables, if any, their domains haven't changed and considered as active values. The abstract CFG is shown in Fig. 4.

![Fig. 4. Reduced CFG of program in Fig. 2.](image)

Observed techniques provide good framework for reducing the size of the model. Reduced model are much smaller and could be verified with much less computational power then the full ones.

The work was carried out within the framework of the international double degree Computer Science Masters program between Politecnico di Milano University (Italy) and Moscow State Institute of Radioengineering, Electronics and Automation on the basis of Integrated Circuits, Nanoelectronics Devices and Microsystems Design Center.

**REFERENCES**