ISBN 978-981-11-3671-9
Proceedings of 2017 the 7th International Workshop on Computer Science and Engineering (WCSE 2017)
Beijing, 25-27 June, 2017, pp. 366-370
doi: 10.18178/wcse.2017.06.063

Formal Verification for AltaRica3.0 Models Based on SPIN

Hu Jun[1+], Li Wanqian [1], Wang Mingming [1] and Zhang Weijun [1]
[1] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, 210016, NanJing, China

Abstract. AltaRica is a kind of modeling language for safety critical systems, AltaRica3.0 is its latest version. There is still no corresponding formal verification method to analyze and verify the AltaRica3.0 model. The main work of this article is to analyze the AltaRica3.0 model by using a model test tool SPIN to analyze the different characteristics of AltaRica3.0 in relation to the previous version. Considering the basic structure of the underlying model GTS, based on the idea of AltaRica3.0 flattening to a GTS model, the core conversion rules of the AltaRica3.0 model to Promela model and a conversion framework are proposed. Based on the case analysis of the wheel brake system (WBS) in civil aircraft, the AltaRica3.0 model was established, and the Promela model was generated by the conversion rule. According to the safety requirements of the wheel brake system in 4761, SPIN is used to verify the system security attributes formally.

Keywords: safety-critical system, AltaRica3.0, SPIN, Wheel Brake System.

1. Introduction

A appropriate safety assessment technique is required in the field of safety critical systems. Due to the sharp increase in the complexity of compter systems, the method of Model Based Safety Analysis(MBSA) gets much attention in the industrial field in recent years.

AltaRica[1] is an MBSA modeling language designed specifically for the safety assessment of industrial systems by the French industrial and Academic Association[2]. AltaRica has elevated to the third generation of AltaRica3.0[3]，which added support for the circulatory system and non-causal systems[4]，increasing the application fields significantly. Currently there is no evaluation software which can provide perfect support for AltaRica3.0. SPIN[5] which have been widely used in the safety analysis and evaluation software can generate counter examples of validated models that are not passed for further analysis and evaluation to further analyze and evaluate the model. Due to the input model of SPIN is Promela, it is necessary to convert the AltaRica3.0 model into the Promela model[6]. This article provides a convertion method from AltaRica3.0 to Promela model.

This article is organized as follows: chapter two introduces the basic concept of AltaRica3.0 language, chapter three introduces the conversion method from AltaRica3.0 language to Promela language, chapter four shows the example of modeling and transiting Wheel Brake System[7] in the avionics system, making safety analysis and verification with SPIN according to its own characteristics.

2. AltaRica3.0 Language

The AltaRica language is a modeling language that describes constrained automata, and each component is described as a constraint automaton. AltaRica language development has a total of three versions, namely AltaRica Data-Flow1.0, AltaRica Data-Flow2.0 and AltaRica3.0[4]. In the version of AltaRica Data-Flow, the model is based on data flow and does not support systems with loop and non-causalty. AltaRica3.0 is based on GTS (Guarded Transition Systems), adding the support for loop by calculating FixPoint to maintain the balance of the variables in the loop. Relative to the previous two versions, AltaRica3.0 can be used to

[+] Corresponding author. Tel.: 13952046531; fax: 86 25084896848.
E-mail address: hujun.nju@139.com.

describe the circulatory system, but there is no relatively perfect verification tool. AltaRica3.0 model is divided into four parts, variable definition, event definition, transition and assertion.

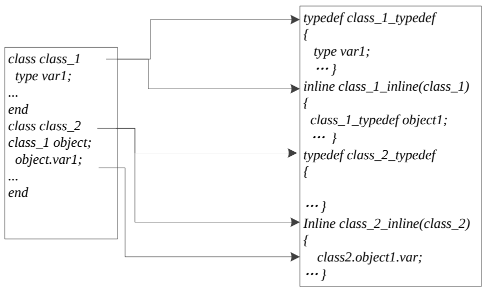
3. Transformation from AltaRica3.0 to Promela

Promela is a modeling language for concurrent systems and an input model for the exhaustive validation tool SPIN. SPIN tools can perform exhaustive validation of the model. In addition, the SPIN tool can simulate and validate the model with Linear Temporary Logic formula. It can display the state space of the model, the size of the state space, the number of conversions, the depth of access and so on. The model can also generate a counter example to modify the model. This section will describe the transformation from AltaRica3.0 language to Promela language. Since the proposed method is based on the idea of AltaRica3.0 flattening to a GTS model, and the implementation of the GTS model is serial, so it only call the initialization process when it is converted to Promela rather than create a child process, that means it uses a single-process Promela model. Following will point out the several difficulties of conversion method.

3.1. Transformation of Class/Block

In the AltaRica3.0 model, the class and block are abstracts of the component. The difference between them is that the block is a solid node and does not need to be instantiated before it can be used and can not be multiplexed. But a block component can be embedded in another block. The class is an abstract node that needs to be instantiated before it can be used. It can't embed in entity nodes, but can be instantiated by multiple objects. According to object-oriented ideas, classes and blocks are actually classes that encapsulate data and operations, so when translated to Promela, the typedef of package data and the inline of the encapsulation operation are used to represent classes or blocks.

The internal of class/block in AltaRica3.0 is generally divided into three parts, namely the variable definition (including event definition), transition and assertion. According to the underlying GTS model, it is known that the class will initialize the variable and then have the assertion. If the event occurs, it will first trigger the transition, then implement the assertion. Otherwise will direct the end. So when translated to Promela, the inline can be divided into five parts, namely variable definition, initialization assertion, event trigger decision, transition execution and the second assertion operation. Depending on whether the event is triggered or not, the execution will be different. The specific transition process is shown in Figure 1.



```
class class_1
  type var1;
...
end
class class_2
class_1 object;
  object.var1;
...
end
```

```
typedef class_1_typedef
{
   type var1;
   ··· }
inline class_1_inline(class_1)
{
  class_1_typedef object1;
  ···  }
typedef class_2_typedef
{

 ··· }
Inline class_2_inline(class_2)
{
   class2.object1.var;
 ··· }
```

Fig.1: The transition of class

3.2. Transformation of FixPoint

AltaRica3.0 on the loop processing is done through the FixPoint. The system puts operations associated with all the flow variables on the loop into a FixPoint, and then cycles through this operation until all the values of flow variables on the loop do not change. Therefore the Promela loop operation can be very good to complete this operation, which cycle implementation on all flow variables operations in the loop and the exit cycle of the loop when all the flow variable values are unchangable.

3.3. Transition and Trigger of the Event

The occurrence of the transition is triggered by the event in AltaRica3.0. Promela language does not have the concept of event, where the enumeration type (mtype) is selected to define the event variable. Since multiple enumeration types are automatically merged into an enumeration type by the system in the Promela, the event variables for all components in AltaRica3.0 are defined as enumeration values in the same enumeration type. Because the enumeration values in the enumeration type can not be repeated, the duplicate enumeration values are removed. In addition, specify that the enumeration value "ZERO" indicates that no event is triggered and its corresponding transition is jumped to the last of inline.

Each class/block can have multiple events in AltaRica3.0, but each time the maximum of implementation is only one event (multiple events of the synchronization event is considered an event), and also can not trigger any event. After the event is triggered, if the guard is satisfied then perform the appropriate transition and implement the assertions, if not then get over. In Promela we use the if statement to simulate the trigger of events in AltaRica3.0. There are a number of statements with "::" mark in if-fi structure of Promela language. Each statement is composed by the guard and the implementation (if ignore the guards then take them as true). When implement the structure of if-fi, we will select the statement which the guard is true from the statement with the mark of '::'. And if there are multiple, one of them is randomly executed. Thus, Promela will select randomly among the statement with the mark of '::' and without guard which can simulate the trigger of event. The trigger of the event is shown in Figure 2.

3.4. Transformation of Synchronous

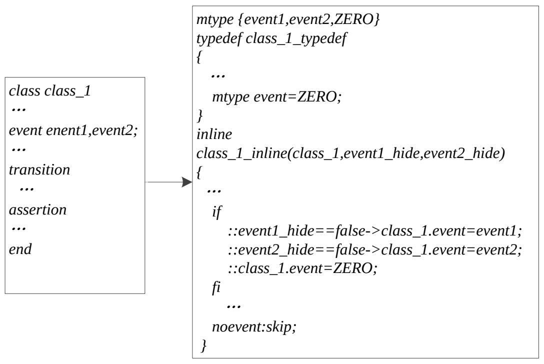
Similar with the processing of synchronization in AltaRica3.0 flattening into GTS model, transformation of synchronous combines several synchronization events into a synchronization event, and then hides the event marked with 'hide' from the model. Synchronization is divided into strong synchronization, weak synchronization and CCF synchronization, the following describes the transformation process:

(1) Strong synchronization: Let two strong synchronization events be e1 and e2, which transitions are e1: g1-> t1 and e2: g2-> t2. Then the strong synchronization event e:! E1 &! E2 is flattened into E: g1 & g2 -> {t1; t2;}, then translate e as a normal event according to the method described in 3.3 to Promela statement. Multiple events of the strong synchronization are the same.

(2) Weak synchronization: Let two weak synchronization events be e1 and e2, which transitions are e1: g1-> t1 and e2: g2-> t2. Then the weak synchronization event e:? E1 & e2 is flattened into e : G1 | g2 -> {if g1 then t1; if g2 then t2;}, then translate e as a normal event according to the method described in 3.3 to Promela statement. Multiple events of the weak synchronization are the same.

(3) CCF synchronization: CCF synchronization is based on weak synchronization and have no hide mark. Its transition method is the same as weak synchronization.

To fully describe the synchronization of the semantics will use the hide keyword, the event with the keyword hide will not be compiled. That means the event with hide keyword will not be triggered. In order to implement this functionality in the Promela language, you may wish to add variables to the list of parameters in the inline to indicate which events are hidden which events are not hidden. Because hide events often occurs when the upper component calls the underlying component, so whether or not to hide is decided by its caller. Specifies how many bool variables are added to the parameter list in the number of events that correspond to the events, and use these variables to indicate whether the event is hidden, true indicates that event is hidden, false indicates that event is not hidden. Then add judgment in part of the trigger of the event of inline, if an event is hidden, then the event will not be triggered. The specific transition process is shown in Figure 2:



```
class class_1
…
event enent1,event2;
…
transition
  …
assertion
…
end
```

```
mtype {event1,event2,ZERO}
typedef class_1_typedef
{
   …
   mtype event=ZERO;
}
inline
class_1_inline(class_1,event1_hide,event2_hide)
{
  …
   if
      ::event1_hide==false->class_1.event=event1;
      ::event2_hide==false->class_1.event=event2;
      ::class_1.event=ZERO;
   fi
      …
   noevent:skip;
 }
```

Fig. 2: The trigger of event and conversion of synchronization

4. Case Analysis of Wheel Brake System

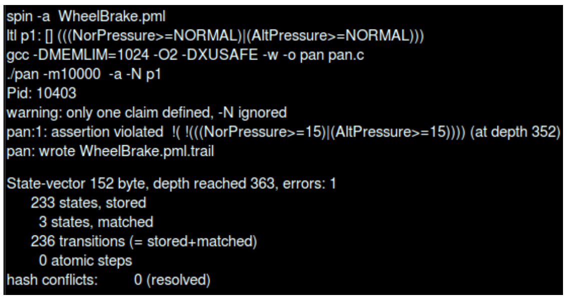
This section uses the design rules to verify the safety of the Wheel Brake System in the avionics system. Firstly, the AltaRica3.0 model is established according to the description and requirements of the system. Then, the AltaRica3.0 model is transformed into the Promela model. Finally, validate the Promela model through the model detection tool SPIN and the safety requirements which satisfied by Linear Temporary Logic (LTL) protocol formula and analyze the resulting validation results.

4.1. Introduction of the System

Wheel Brake System (WBS) is a typical system given in the Civil Aviation Standard SAE ARP4761[7]. The wheel brake system is mounted on the main landing gear of the aircraft, and the pressure on the main wheel is given a minimum value to stop the aircraft during the landing, taxiing and other stages of the aircraft. Each component is regulated to ensure that the final pressure value is greater than the threshold. The whole Wheel Brake System is divided into Brake System Control Unit (BSCU) and Hydraulic Subsystem.

4.2. Verification and Analysis of Wheel Brake System

Modeling the Wheel Brake System with AltaRica3.0, then converting the model to Promela using the method in Chapter 3, and finally using SPIN to validate and analyze the model. To ensure the safety of the Wheel Brake System, it is necessary to require a hydraulic value which on the wheel to be greater than the threshold. It requires that at least one hydraulic value of the normal or spare lines is greater than the threshold[8]. The safety described by LTL is as follows: P1 := [](NorPressure>= ThresHold) |(AltPressure>= ThresHold). Where NorPressure is the hydraulic value of normal line, AltPressure is the hydraulic value of spare line, ThresHold is the threshold value. Verify the Promela model of the Wheel Brake System as shown in Figure 3:

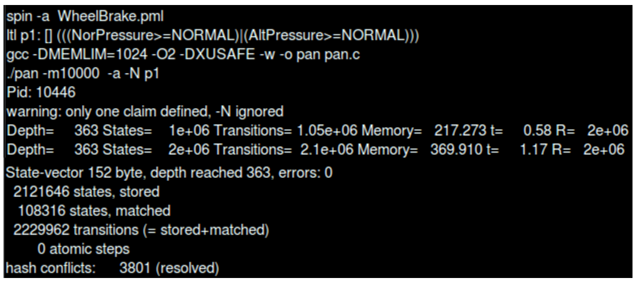


```
spin -a  WheelBrake.pml
ltl p1: [] (((NorPressure>=NORMAL)|(AltPressure>=NORMAL)))
gcc -DMEMLIM=1024 -O2 -DXUSAFE -w -o pan pan.c
./pan -m10000  -a -N p1
Pid: 10403
warning: only one claim defined, -N ignored
pan:1: assertion violated  !( !(((NorPressure>=15)|(AltPressure>=15)))) (at depth 352)
pan: wrote WheelBrake.pml.trail

State-vector 152 byte, depth reached 363, errors: 1
      233 states, stored
        3 states, matched
      236 transitions (= stored+matched)
        0 atomic steps
hash conflicts:         0 (resolved)
```

Fig. 3: Validation results before model modification

The result of the verification indicates that the LTL rule has not been verified. After analyzing the model, it is found that the failure of the CMD / AS component may cause no output value when the system is in the normal line, and the CMD / AS component is under the level of selector valve, its failure can not lead to the selector valve to switch the system to the spare line, so it will lead to the failure of the entire system. When the system is in the spare line, if the failure of blue hydraulic pump and the accumulator pump at the same time will lead to no additional hydraulic pump to provide hydraulic value, resulting in the failure of the entire system. In addition, ValueNoPressure in the command set adjusts the hydraulic value directly to zero, resulting in failure of the entire system. If the current model is slightly modified, remove the failure of CMD / AS limit valve and the accumulator pump, and remove the ValueNoPressure command from the command set to get a new model to import the new model and LTL rules into iSpin to verify. The entire verification process lasts 15s. The results are shown in Figure 4:



```
spin -a  WheelBrake.pml
ltl p1: [] (((NorPressure>=NORMAL)|(AltPressure>=NORMAL)))
gcc -DMEMLIM=1024 -O2 -DXUSAFE -w -o pan pan.c
./pan -m10000  -a -N p1
Pid: 10446
warning: only one claim defined, -N ignored
Depth=    363 States=    1e+06 Transitions= 1.05e+06 Memory=   217.273 t=     0.58 R=   2e+06
Depth=    363 States=    2e+06 Transitions=  2.1e+06 Memory=   369.910 t=     1.17 R=   2e+06
State-vector 152 byte, depth reached 363, errors: 0
  2121646 states, stored
   108316 states, matched
  2229962 transitions (= stored+matched)
        0 atomic steps
hash conflicts:      3801 (resolved)
```

Fig. 4: Validation results after model modification

The rule is validated within valid time, indicating that it is a failure of the key component and the ValueNoPressure command which cause the failure, then indicating that there is a problem with the model currently being established. In fact, some of the details of the Wheel Brake System in 4761 are not described in detail, so many of the details of the modeling are by virtue of the understanding. According to the verification results, we can know that the perfect direction of the model will be removing the hydraulic value directly from the instruction set, adding the corresponding mechanism for the key module to improve the safety, or setting the redundancy to improve the safety.

5. Conclusion

In this article, a conversion method from AltaRica3.0 to Promela model is proposed and proved formally. Use the Wheel Brake System in SAE ARP4761 to establish the AltaRica3.0 model and transform it to Promela model. Finally, Spin is used to analyze and verify the modified Promela model, the problems existing in the model are found and the modification of the model is put forward.

The future work mainly includes two aspects: (1) Modify and improve the model to make it more in line with safety standards for exposed problems of the Wheel Brake System model. (2) Improve the conversion method, and through the preparation of software to transform the models automatically.

6. Acknowledgements

This work is Supported by: The National Basic Research Program of China (973 Program) (2014CB744903); Funding of Jiangsu Innovation Program for Graduate Education (SJZZ16_006), the Fundamental Research Funds for the Central Universities.

7. References

[1] Arnold, André, et al. "The AltaRica formalism for describing concurrent systems." *Fundamenta Informaticae* 40.2, 3 (1999): 109-124.

[2] Bieber, Pierre, et al. "Safety assessment with altarica." *Building the Information Society*. Springer US, 2004. 505-510.

[3] Prosvirnova, Tatiana, et al. "The altarica 3.0 project for model-based safety assessment." *IFAC Proceedings Volumes* 46.22 (2013): 127-132.

[4] Batteux, M., T. Prosvirnova, and A. Rauzy. "AltaRica 3.0 Language Specification. November 2015." *Altarica association report* (2015).

[5] Clarke, Edmund M., Orna Grumberg, and Doron Peled. *Model checking*. MIT press, 1999.

[6] Inverardi, Paola, Henry Muccini, and Patrizio Pelliccione. "Automated check of architectural models consistency using SPIN." *Automated Software Engineering, 2001.(ASE 2001). Proceedings. 16th Annual International Conference on*. IEEE, 2001.

[7] SAE International. *Guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment*. SAE International, 1996.

[8] Joshi, Anjali, et al. "Model-based safety analysis." (2006).