Difference compression in SPIN

Benoît PARREAUX

Laboratoire d'Informatique de Besançon
Université de Frache-Comté
Route de Gray
25030 Besançon Cedex

E-mail: parreaux@lib.univ-fcomte.fr

SPIN 98 WORKSHOP

Summary: *One of main problems induced by the technique of model-checking is state explosion. Some work aim to improve the rate of covering the graph of accessibility. Different approaches have been proposed : bitstate hashing [Hol 95], partial order[God 96a], collapse mode [Hol 97], methods based on BDD [God 96b] [Mil 93] and compressing state vectors[Hol 92] [Vis 96].*

Our work originates from this problem. We propose a new compression method using a hash-table. This technique comes from methods for processing animated images. The state compression is done in two steps. Given a hash-code, the first vector using it is chosen as reference vector. First, we compute the difference between the state vector and the reference vector. Second, we compress the resulting vector using a RLE-like technique. If a more sized vector is generated, the reference vector is enlarged with additional bits. Modifying the reference vector this way preserves the method's reversibility since a vector is always compressed with respect to the same part of the reference vector and always yields to the same compression even if the reference size is increased. This reversible technique can be viewed as an exhaustive way to put model-checking into action.

We get compression rates close to other classical methods with a quicker graph traversal. In addition, some improvements could be done.

Then we sketch three possible improvements. Significant improvement can be got with using a suitable hash function, e.g., a hash function that chaining elements including identical subparts. Second, a dynamic choice of the reference vector would result in increasing the compression rate. For example, the reference vector could be chosen among several reference vectors for a collision chain. A third improvement would be to modify the implementation so that it becomes compatible with the collapse mode.

Key words: Model Checking, Compression and memory storage.

1.INTRODUCTION

The technique of model checking [Ger 95] allows verifying the communicating process behavior by realizing all possible rendered executions by the model and by verifying properties on these states. During this verification a graph, called graph of accessibility, is created. The former contains all states already meet. This type of verification, told exhaustive, demand a lot resource memory [Cou 92]. It is necessary therefore to use the former at best so as to obtain from rate goods of cover. Many works of research have been realized so as to increase the number of accessible states [Hol 97]. Especially all a series of works have tackled the problem by decreasing the size of the representation of states in memory. Many specific or traditional techniques to the problem that one seeks to solve have been presented and tried. We register our called technique compression by difference in this last. Different techniques of compression have been used to reduce the size of the representation of states in memory. Some of these techniques as the bitstate hashing [Hol 95] [Maj 96] are not reversible. These methods are very efficient in term of compression because their no reversibility is the consequence of the loss of a part more or less important of the information contained in the vector of state. This type of compressions does not allow knowing if the whole of the graph of accessibility has been reached. The obtained result is therefore partial since one can not pull conclusions that if an error has been found. Others methods have not this defect and are therefore reversible although less efficient. On the other hand these techniques allow always concluding when it ends. Among the former, one can find traditional compression methods as the byte masking, compression RLE or Huffman, or specific as the Collapse has brought into play. These methods, in addition to the reversibility, have all a property in common: it is necessary that the compressed representation of a state remains the even all along the execution, to allow the comparison of states. Techniques resting on dictionaries, such that Huffman, are little efficient due to the fact the difficulty to realize an optimum dictionary for all problems. These are these reasons that have given birth to two passes methods. The first passes server to create the dictionary, the second realizing the verification by using this dictionary to code vectors of state. An alternative to traditional methods has been found by means of the utilization of the Collapse. This technique implanted in SPIN gives very good results. In the paragraph following we are going to study methods of compression reversible. We explain director ideas of these algorithms to finish by concluding on advantages and disadvantages of these techniques. In the third paragraph appears our proposal. We begin by presenting underlying ideas and reasons that us have elaborate to develop this technique. We give some implantation details of this technique. We finish in the fourth paragraph by concluding by giving obtained results and by releasing some perspectives to these works.

2. EXISTENT COMPRESSION TECHNIQUES

Two ways have been followed in the choice of compression algorithms. The first has been implemented of already known general algorithms. We will quote three techniques: the bit masking, the Run-Length encoding and the method of static Huffman. We will speak to finish the mode Collapse introduced in the version 2.7.4 of SPIN. This technique is interesting because it is specific to properties of state vectors generated by model checking. It pulls indeed left specificity of state vectors to decrease the size of their representation.

The simplest technique, the byte masking [Hol 92] belongs the distributed version of SPIN since several years. This method of compression functions by localizing in the vector of byte state that are "known" to contain constant values. One can quote for example:

- bytes of filling added by the compiler in order that the size of data structures are a multiple of that a word,
- structures of data used for implement Rendez-Vous canals. Constant parts are marked with a byte Mask.

When vectors of state are memorized, alone bytes marked not are copied and others are ignored. To warn the confusion states during the comparison, the original length of each vector of state is prefixed to its compressed version. In the case where the vector of state would be less great than 216 bytes one adds to the most two bytes to the vector of compress state (four for values greater). These operations of "masking" are reversible what guarantee that the compression is without loss. This method allows a moderated reduction of the place used in memory for a weak cost to the level of the execution.

Other solution is to apply the technique of the Run-Length Encoding [Hol 92] on the vector to compress. In all experiences proposed the size of the vector to stock is a multiple of the byte. In the method of the Run-Length Encoding, the vector of state is coded as a sequence of byte pairs. The first part of the pair gives the number of time that the repeat value of the byte. The second part gives the original value of the byte. Clearly this method is efficient solely if on the average each byte appears at least twice consecutively in vectors of state. If this is not the case this method can bring an increase of the size of the necessary memory to stock the vector of state until twice. This method has for results to increase significantly the time of execution without for as much give good results to the level of the economy of memory.

A last solution consists in use a method of compression of static Huffman [Hol 92] [Knu 73]. To apply this method a statistical study of the composition of state vectors is necessary. These values will be used to create the dictionary of the method of coding by attributing to bytes them most commons, the smallest codes. Bytes are then stocked in a vector whose length can vary. Realized some simulations show us that one obtains thus from best results that with the preceding method for a cost in lesser execution time. These last remain however means that is especially had in the fact that the dictionary is static and define before the execution.

From September 95 in the version 2.7.4 of SPIN a new mode of experimental compression has been introduced to exploit recursive indexation methods [Hol 96] [Vis 97] for the stocking of state vectors. The method is based on the verification that the complexity of the asynchronous system verification comes mainly from the no determinism during the execution of process. Each process and each datum takes only a relatively small number of different values. The number of combinations that one can realize from these local states obtains the very great number of accessible states of the system. To repeat the description completes local components for each global vector is therefore very expensive in memory and efficient bit. In the first mode of compression Collapse, the each local component stocking of the vector, is realized by coding local states represented by each process or canal separately in a hash table clean and it is appointed it a unique forefinger number. These are these numbers that form then the global state vector.

A defect in the first implantation of this method is that the user has to define a maximum for the greatest possible forefinger for components of state vectors. The limit serves to determine if forefinger of components of the vector can be stocked on one, two, three or four bytes. The necessity to give an estimation of the size of the vector of state has prevented to use the former as SPIN compression method by default. A solution to avoid the necessity of specify a upper bound for the number of bytes used by forefinger to stock the state vector components is to stock the number of components and the number of bytes used for each components index. These numbers can now be authorized to change between vectors of state, because the exact compression method or decompression can be always determined of unique manner for each vector. The number of bytes used to code an index never is smaller than a byte and greater than four bytes. To stock this information, only two bits by components are necessary. In this news implantation the index table index size is added to the global variable component. This vector now contains:

- a local index, between one and four bytes, for each active process, that identifies components that contains this state, including all these variable places, but excluding declared canals locally;
- a global index, between one and four bytes, for unique component that contains all shared variables, all canals of message and the table of index weights of all local indices;
- a byte that specify the number of bytes used for the global index.

This technique is called recursive indexation. Traditional techniques as RLE and Huffman are little efficient in look of the time spent for the compression during the verification. A solution to improve results has been the introduction of a first passes in the verification allowing to constitute the dictionary for the compression, then a second pass that uses this dictionary to realize the compression itself. The collapse mode gives good results without increasing excessive manner the time of calculation. This technique uses mainly the property of state vectors that makes that the former are, by part, identical some with others.

The technique that we have developed and that we present in the next paragraph uses this property.

3. OUR PROPOSAL

We are left from the principle that a technique adapted to the problem is more efficient than a general technique. In order that it is necessary to study vectors of state to try to find properties. A characteristic of state vectors is that two consecutive vectors on a path of execution are in general little different. It would be therefore economic to the level the memory to code a vector thanks to its difference with the before vector during the verification. This type of coding is not possible because the same vector of state could have several memory representations according to the manner whose it has been reached. That does not invite to our problematical since it would be necessary each time that a state is generated, to know is there exists already, reconstruct all already found states. That would increase unacceptable manner the time of execution. It is necessary therefore that the method associates with each vector of state an alone memory representation.

We have then thought of code state vectors them some by report to others in chains of collision of the hash table. A state is therefore coded unique manner since the hash code of the former is unique.

3.1. Presentation

To render the more rapid coding, we used a vector by collision chain with that will be made differences with all the other vectors. We will call the former reference vectors and they are regrouped in what we will appoint basis. The former is constituted under way of execution and these elements once known does not have to change value. If a vector of basis changes, it is necessary indeed recalculate all representation memory of state vectors in the relevant collision chain. Each vector of the graph of state will be coded according to its difference to one of the vectors of the basis and an alone. The rate of compression will be linked to the composition of collision chains of the hash table. More elements of these chains will be similar, more results obtained in term of compression will be good. The same manner it is necessary that there is more an element in the chain of collision in order that the compression functions. It is necessary us to explain simply what we hear by coding a state by difference. We are going to begin by making its difference bit to bit as compared to a state of reference. One will take for state of references the first state met in a collision chain. This state will be stocked without being compress as first element the hash chain. It is necessary therefore that this element remains constant all along the verification in order that states are always coded of the same manner.

Once the difference bit to bit realized we packed continuations of '0' obtained to reduce the size of the vector of state. Several verifications can be suitable:

- It will be necessary to have a clamping of more of an element in the collision chain in order that the method has results.
- More the clamping is long less the research of elements and efficient.

- The function of calculation of the difference will have for good performances to be linked to the hash function.
- The function of the difference bit to bit calculation is reversible.

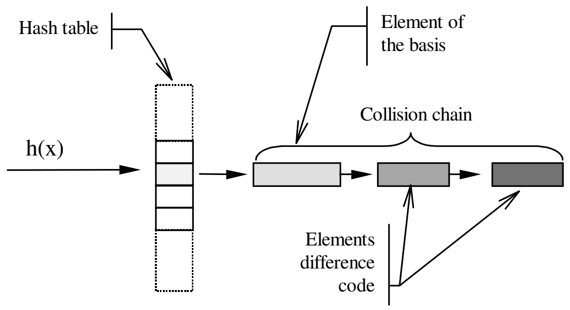

Figure 1 : Compression by difference.

We have therefore studied hash functions to find that that will be the most adapted to our problem. In order that an hash algorithm has been implement in C or C and will allow to test the different possible solutions. Different means have been tested, different functions with different values of parameters. Tests have been realized thanks to a version to modify SPIN that has allowed us to determine if our idea was valid. Results have been very convincing, even without modifying the initial hash function of SPIN. This conclusion us have therefore pushed to continue our experience. A more elaborate study of usable functions as hash function could have improved results. This study have not succeeded to lead it sufficiently long and with sufficiently success to obtain a function giving results superior others. We have tested different classic functions that have all result data equal. Of course some were more effective for some problems but less in others. We think however than a more thorough study of the definition of these functions would allow to render the technique again more efficient.

One of the problems of this technique is its incompatibility with the Collapse mode. More precisely if one uses this technique it is impossible then to realize the Collapse. Indeed, if the difference preserves the different parts of the vector, the compression of the '0' that one realizes then no longer allows to stock on the one hand the local part and the other the global part. It could have however be envisaged to realize the difference independently on each part of the vector of state and therefore after having realized the Collapse.

We go to finish to study how the implantation of this technique has been realized in practice and results obtained.

3.2. Implantation and results

We have chosen implant this method in SPIN. Our choice is it has several factors: the availability of sources, a work [Hol a] documented perfectly these sources, a certain masters the tools established during different works (for example [Jul 98a]). More this platform has served as basis to bring different techniques into play seen some to earn the place in memory. That allows us more easily to verify the efficiency of our method by testing it on problems used to realize others performance studies.

A hash function is a function that has for image area an integer interval. This value represents then the position in a table of bits. To insure that the function returns a whole belonging to given interval as result, it is current to use the mathematical function mod(N). All other function having for image a finished area corresponding to that waited is susceptible to suit. Other function is generally used just before this last for the purpose to distribute the most efficiently possible elements in the table. One uses often a polynomial but nothing prevents to use other types of function. It is elsewhere on this last that we have tempted to intervene to increase the efficiency of the method.

The implantation of the method has been realized with Stéphane Michaut studying in DESS. We have studied the code of SPIN in detail to find the function in which was realized the detection of collisions and the addition of the new states vectors in the hash table. Once the function realizing that found, it has been necessary to modify the code so as to realize the difference and the compression of '0' chain. We have written a function realizing the difference bit to bit between the vector running and the vector of basis. This difference is realized by the utilization of a XOR between the two vectors. An other then '0' chain compress by using a coding resembling to that use for RLE but simpler and more rapid. It has then been necessary to verify that these two-chained operations were reversible. We have therefore compressed and decompress different vectors to verify that they were not modified. These operations being reversible we have begun by looking what would be results without truly changing the internal functioning of SPIN. We have therefore calculated the size of vectors before compression and the size of vectors after compression more the size of vectors of the basis. We have then recorded good rate of compression (20 to 60%) for times of different little execution. However we have outline that when the size of the hash table decreased the rate of compression decreased also. After a small study have observed that that was the to what vectors of state choose as fundamental were too small. The difference with vectors following when they were great size did not allow showing sufficiently '0' in order that the technique is effective. It us has also seemed that the same problem could have occur when we would generate a great space of state. The solution that we have found in this problem is to increase the size of the vector of basis when one finds a vector of state greater. One adds then to the vector of bases the bits supplementary of the vector of state running.

The compression remains reversible because for a given size vector, one is on that the vector of basis reduced to this size remains constant. With this result modification are almost identical in the best of cases but degrading far more slowly with the increase of the ratio Number of states / hash table size.

4. CONCLUSIONS AND PERSPECTIVES

We have obtained some results. The obtained gain varies between 20 and 60% according to problems processed. These results have been obtained without touching the hash function. The obtained gain varies in function of the problem processed, especially more the number of states is important, best is the rate of compression. That explains by the fact that in the second case the number of collisions is more important. The technique of compression using the former, this result is not surprising. When the number of states is little important a solution is to reduce the size of the hash table. That has indeed a very good influences on results obtained the gain is often important enough. However, put aside some very specific problems the obtained maximum gain situates around 50%.

We think interesting in the future to work on the hash function used. We think indeed that by improving this function of results again more interesting could have been obtained. It can also seem interesting to implant the compatible variant with the collapse mode. A possible solution is realized in a first time it collapse then to practice the compression by difference independently on each part of the vector. A light adjustment of the method would allow obtaining this result. It us has not however been possible to realize necessary modifications in the code of SPIN to be able to test this option and to compare results obtained to these of the other methods.

Others works are to lead in this direction. It seems indeed that a function of very simple compression working on data pretreated gives good results so to the level of times of compression, that to the level of gains obtained.

5. BIBLIOGRAPHY

[Cou 92] C. COURCOUBETIS, Mr. VARDI, P. WOLPER, Mr. YANNAKAKIS, "Memory efficient algorithms for the verification of temporal properties", Formal Methods in System Design I, pp 275-288, 1992.

[Ger 95] R. GERTH, D. PELED, Mr. VARDI, P. WOLPER," Simple one - fly automatic verification of linear temporal logic", Proc. PSTV95 Conference, Warsaw, Poland, June 1995.

[God 93] P. GODEFROID and G.J. HOLZMANN, "One the verification of temporal properties", Proc. IFIP / WG6.1 Symp. One Protocols Specification, Testing and Verification, PSTV93, Liege, Belgium, June 1993

[God 96a] P. GODEFROID, "Partial order Methods for the verification of the Concurrent Systems", LNCS Vol. 1032, Springer Verlag, 1996.

[God 96b] P. GODEFROID, "Symbolic protocol verification with Queue BDDs", Proc. Logic in Computer science, Rutgers Univ., New Brunswick, New Jersey, July 1996.

[Hol 91] G.J. HOLZMANN, "Design and validation of protocols", Prentice entrance software series, 1991.

[Hol 92] G.J. HOLZMANN, P. GODEFROID and D. PIROTTIN, "Coverage preserving reduction strategies for reachability analysis", Proc. 12th Int. Conf. On Protocol Specification, Testing and verification, PSTV 92, Orlando, FI., 1992.

[Hol 95] G.J. HOLZMANN, "An analysis of bit state hashing", Proc. IFIP/WG6.1 Symp. on Protocol Specification, Testing, and Verification, PSTV 95, Warsaw, poland.

[Hol 96] G.J. HOLZMANN, "The model checker SPIN", IEEE Trans. One Software Engineering, Flight. 23, n° 5, 1996.

[Hol 97] G.J. HOLZMANN, "State Compression in Spin", Proc. Third Spin Workshop, Twente University The Netherlands, April 1997.

[Hol a] G.J. HOLZMANN, "Basic Spin manual", Bell laboratories - Murray hill, NJ 07974.

[Hol b] G.J. HOLZMANN, "Overview of the Spin model checker", Technical Transfer, Computing Principles Research Department, Bell Laboratories.

[Jul 98a] J. JULLIAND, B. LEGEARD, T. MACHICOANE, B. PARREAUX, B. TATIBOUET, "Specification of year integrated circuits card protocol application using B and Temporal Linear Logic", 2nd Conference one the B method, Montpellier, April 1998.

[Knu 73] D.E. KNUTH, "The Art of Computer Programming", Vol. 1 Addison-Wesley, 1973.

[Maj 96] B.S. MAJEWSKI, N.C. WORMALD, G. HAVAS and Z.J. CZECH, "A family of perfect hashing methods", The computer Journal, Vol. 39, No. 6, pp 547-554.

[Mil 93] K. McMILLAN, "Symbolic model checking", Kluwer Academic Publishers, 1993.

[Vis 96] W. VISSER, "Memory efficient storage in SPIN", Proc. 2nd SPIN Workshop, DIMACS, Rutgers Univ., New Brunswick, New Jersey, July 1996.