GRIP: Generic Representatives in PRISM

Alastair F. Donaldson
Codeplay Software Ltd.
Edinburgh
Email: ally@codeplay.com

Alice Miller
Department of Computing Science
University of Glasgow
Email: alice@dcs.gla.ac.uk

David Parker
School of Computer Science
University of Birmingham
Email: dxp@cs.bham.ac.uk

***Abstract*— We give an overview of GRIP, a symmetry reduction tool for the probabilistic model checker PRISM, together with experimental results for a selection of example specifications.**

I. An Overview of GRIP

GRIP (generic representatives in PRISM), introduced in [1], is a symmetry reduction tool for the PRISM model checker [6]. GRIP is based on the *generic representatives* approach of [2], which aims to overcome the inherent problem of combining symmetry reduction with symbolic state-space representation. We present an overview of GRIP version 2.0 (referred to henceforth as GRIP), an improved version of the original tool, and compare GRIP to PRISM-symm, an alternative symmetry reduction tool for PRISM [5]. GRIP, together with the PRISM examples used for experiments in Section III can be downloaded from our website [4].

The top panel of Figure 1 shows a simple leader election protocol in PRISM, adapted from [1]. The underlying model here is a Markov decision process (MDP). GRIP works by translating this specification into a *reduced* form, as shown in the bottom-left panel of the figure. The reduced specification abstracts away from specific modules, instead using a single *generic* module comprised of variables which *count* the number of modules in each potential local state. Symmetric temporal properties can also be translated into reduced form. PRISM can then be used, unchanged, to check reduced properties of a reduced specification.

II. New Features of GRIP

The original version of GRIP required specifications to consist of multiple instantiations of a single symmetric module type, specified using a single local state variable. This model of computation is in keeping with the presentation of the generic representatives approach for non-probabilistic model checking [2]. While a wide class of symmetric systems can, in theory, be specified in this way, accurately modelling complex protocols via a single state variable quickly becomes impractical.

GRIP now supports: multiple local state variables; a wide range of arithmetic and boolean expressions over these variables; communication via shared global variables, and multiple asymmetric modules in parallel with a single family of symmetric modules. In addition, GRIP handles models with continuous time Markov chain (CTMC) semantics.

Supporting multiple local variables results in a large number of potential local states, which translates to many counters in the specification output by GRIP. This in turn can lead to large MTBDDs (the symbolic data structure used by PRISM). To combat this, we have implemented an optimisation suggested in [3]: we use PRISM for local reachability analysis during the translation process, to reduce the number of counters in the output specification. In addition, since the sum of counter variables should always equal N (the number of symmetric modules), the last counter variable can be eliminated and replaced with the formula $C_k = N - (\sum_{i=1}^{k-1} C_i)$. This second optimisation offers a modest reduction in MTBDD size. The bottom-right panel of Figure 1 shows the effect of these optimisations: local reachability analysis determines that the local state $(0, 1)$ (where `init`$i = 0$ and `reg`$i = 1$) is unreachable, eliminating the need for the `no_3` variable and associated statements. The `no_2` variable is then replaced with a formula.

III. Experimental Results and Discussion

Figure 2 summarises experimental results for model building with PRISM, PRISM-symm and GRIP (with and without optimisations). We have used five families of models as case-studies, all of which are described in detail on the PRISM website [6]. The *consensus*, *byzantine* and *rabin* models are MDPs; the *fgf* and *peer2peer* models are CTMCs. For reasons of space we have omitted model checking times, but note that all symmetry-reduced models are feasible to model check. In all cases, translation into reduced form using GRIP took less than two seconds. Experiments were performed on a 2.80GHz PC with 1GB RAM.

It is not surprising that exploiting symmetry leads to a large state-space reduction. However, for symbolic model checking it is the MTBDD size for the resulting symmetric model that determines whether the technique is feasible. In this respect, the optimisations implemented by GRIP are clearly effective: optimised GRIP outperforms PRISM-symm in terms of MTBDD size for all of the MDP examples and is comparable for the larger CTMC examples. GRIP also offers an improvement in building time for larger *fgf* models. Note that although both GRIP and PRISM-symm produce larger MTBDDs than PRISM for the CTMC models, the reduction in state spaces make much larger models now amenable to model checking. On the *rabin* examples, GRIP requires longer to build models than the other techniques. This is due to complex expressions which arise in the translated PRISM code.

Input specification

```
module processor1
  init1 : [0..1];  reg1 : [0..1];
  [] (init1=0) -> (init1'=1)&(reg1'=0);
  [] (init1=0) -> (init1'=1)&(reg1'=1);
  [] (init1=1 & reg1=0) & (init2=1 & init3=1) & (reg2=0 & reg3=0) -> 0.5:(reg1'=0) + 0.5:(reg1'=1);
  [] (init1=1 & reg1=0) & (init2=1 & init3=1) & (reg2=1 | reg3=1) -> (reg1'=0);
  [] (init1=1 & reg1=1) & (init2=1 & init3=1) & (reg2=1 | reg3=1) -> 0.5:(reg1'=0) + 0.5:(reg1'=1);
  [] (init1=1 & reg1=1) & (init2=1 & init3=1) & (reg2=0 & reg3=0) -> (reg1'=1);
endmodule

module processor2 = processor1 [ init1 = init2, init2 = init1, reg1 = reg2, reg2=reg1 ] endmodule
module processor3 = processor1 [ init1 = init3, init3 = init1, reg1 = reg3, reg3=reg1 ] endmodule
```

Reduced specification

```
module generic_process
 no_0 : [0..3] init 3; // modules in state (0,0)
 no_1 : [0..3] init 0; // modules in state (0,1)
 no_2 : [0..3] init 0; // modules in state (1,0)
 no_3 : [0..3] init 0; // modules in state (1,1)

 [] (no_0>0) -> (no_0'=no_0-1)&(no_2'=no_2+1);
 [] (no_1>0) -> (no_1'=no_1-1)&(no_2'=no_2+1);
 [] (no_0>0) -> (no_0'=no_0-1)&(no_3'=no_3+1);
 [] (no_1>0) -> (no_1'=no_1-1)&(no_3'=no_3+1);
 [] (no_2>0) & (no_2+no_3=3)&(no_0+no_2=3) -> 0.5:true + 0.5:(no_2'=no_2-1)&(no_3'=no_3+1);
 [] (no_2>0) & (no_2+no_3=3)&(no_1+no_3>0) -> true;
 [] (no_3>0) & (no_2+no_3=3)&(no_1+no_3>1) -> 0.5:(no_3'=no_3-1)&(no_2'=no_2+1) + 0.5:true;
 [] (no_3>0) & (no_2+no_3=3)&(no_0+no_2=2) -> true;
endmodule
```

Reduced specification with optimisations

```
formula no_2 = (3-(no_0+no_1)); // modules in state (1,1)

module generic_process
 no_0 : [0..3] init 3; // modules in state (0,0)
 no_1 : [0..3] init 0; // modules in state (1,0)

 [] (no_0>0) -> (no_0'=no_0-1)&(no_1'=no_1+1);
 [] (no_0>0) -> (no_0'=no_0-1);
 [] (no_1>0) & (no_1+no_2=3)&(no_0+no_1=3)
                -> 0.5:true + 0.5:(no_1'=no_1-1);
 [] (no_1>0) & (no_1+no_2=3)&(no_2>0) -> true;
 [] (no_2>0) & (no_1+no_2=3)&(no_2>1)
                -> 0.5:(no_1'=no_1+1) + 0.5:true;
 [] (no_2>0) & (no_1+no_2=3)&(no_0+no_1=2) -> true;
endmodule
```

Fig. 1. Applying GRIP to a simple leader election specification, with and without optimisations.

N	States		MTBDD (nodes)				Build time (sec.)			
	Full	Symm	PRISM	PRISM-symm	GRIP	GRIP (optimised)	PRISM	PRISM-symm	GRIP	GRIP (optimised)
consensus: shared coin-flipping protocol from Aspnes & Herlihy's randomised consensus protocol										
10	2.8e+9	136,708	29,419	29,939	38,319	24,914	3.25	3.96	4.07	4.71
12	1.2e+11	339,729	50,037	50,741	56,374	36,819	5.65	7.85	7.21	6.95
14	5.0e+12	747,243	78,171	79,123	78,072	51,192	9.44	14.5	11.41	10.2
16	2.1e+14	1,497,972	115,385	116,691	123,743	80,485	19.1	26.8	22.43	17.9
byzantine: randomised Byzantine agreement protocol of Cachin, Kursawe & Shoup										
8	6.4e+8	298,993	713,143	167,587	175,046	167,372	20.8	23.40	16.3	17.1
12	1.0e+13	7,994,813	4,257,996	937,484	681,580	646,455	144.1	169.7	37.1	37.2
16	1.9e+16	1.1e+8	13,306,326	2,949,979	1,986,234	1,874,953	975.5	1143	157.6	160.1
rabin: Rabin's randomised mutual exclusion algorithm										
5	6,769,448	87312	136,840	257,446	out of memory	123,512	3.89	8.91	-	34.5
6	1.3e+8	356592	206,213	408,291	out of memory	185,943	7.56	16.9	-	46.7
7	2.5e+9	1271328	287,661	587,917	out of memory	261,474	12.5	28.7	-	50.9
8	4.5e+10	4062048	381,184	796,324	out of memory	430,901	20.2	46.2	-	102.7
fgf: simplified version of the FGF (Fibroblast Growth Factor) signalling pathway										
4	216,961	12,397	83,306	153,818	399,078	262,871	2.397	4.74	10.3	7.51
6	9.6e+7	283,360	522,063	1,044,350	1,784,685	1,222,992	47.4	75.2	73.2	46.4
8	4.1e+10	3,996,135	2,080,931	4,114,456	7,344,006	5,119,910	323.4	497.2	420.8	272.7
10	1.7e+13	4.0e+7	6,314,340	12,024,036	18,660,241	13,264,807	2,135	3,028	2,047	1,275
peer2peer: simple peer-to-peer (P2P) protocol based on BitTorrent										
4	1,048,576	52,360	11,941	42,166	84,280	84,280	0.064	0.63	2.44	3.93
5	3.4e+7	376,992	26,266	101,630	157,476	157,476	0.133	1.32	2.79	4.30
6	1.1e+9	2,324,784	40,591	189,704	247,122	247,122	0.269	2.99	3.80	5.29
7	3.4e+10	1.3e+7	54,916	306,123	355,721	355,721	0.516	5.64	4.63	6.14

Fig. 2. Experimental results for model building using PRISM, PRISM-symm, GRIP and optimised GRIP.

Despite our improvements to GRIP, PRISM-symm can be applied to a wider variety of examples where modules communicate via synchronisation labels. This restriction means that GRIP cannot handle certain case-studies, such as a CSMA protocol, on which PRISM-symm performs well [5]. Another distinction between GRIP and PRISM-symm is that PRISM-symm tends to out-perform GRIP when applied to a specification consisting of a relatively small number of complex modules, whereas GRIP wins out when applied to a large number of simpler modules.

On the other hand, an important advantage of GRIP is that, unlike PRISM-symm, there is no need to first construct the full *unreduced* model. Further, since GRIP merely acts as a pre-processor for PRISM specifications, it automatically provides symmetry reduction for model checking tools which use the PRISM input language, e.g. Ymer, or an input language into which PRISM specifications can be translated, e.g. MRMC (see [6] for links to these tools).

To improve performance, we are considering further MTBDD-related optimisations, as well as techniques to reduce the complexity of the output program produced by GRIP. We also intend to provide a formal treatment of our extended generic representatives approach, which also has positive implications for non-probabilistic model checking.

REFERENCES

[1] A. F. Donaldson and A. Miller. Symmetry reduction for probabilistic model checking using generic representatives. In *ATVA'06*, LNCS 4218, pages 9–23. Springer, 2006.

[2] E. A. Emerson and T. Wahl. On combining symmetry reduction and symbolic representation for efficient model checking. In *CHARME'03*, LNCS 2860, pages 216–230. Springer, 2003.

[3] E. A. Emerson and T. Wahl. Efficient reduction techniques for systems with many components. ENTCS 130:379-399.

[4] GRIP website: `http://www.cs.bham.ac.uk/~dxp/prism/grip`

[5] M. Z. Kwiatkowska, G. Norman, and D. Parker. Symmetry reduction for probabilistic model checking. In *CAV'06*, LNCS 4144, pages 234–248. Springer, 2006.

[6] PRISM website: `http://www.cs.bham.ac.uk/~dxp/prism`