

Circus2CSP: A Tool for Model-Checking *Circus* Using FDR

Artur Oliveira Gomes¹ and Andrew Butterfield²

¹ Universidade Federal de Mato Grosso do Sul, Brazil
artur.gomes@ufms.br

² School of Computer Science and Statistics
Trinity College Dublin, Ireland
butrfield@tcd.ie

Abstract. In this paper, we introduce *Circus2CSP*, a tool that automatically translates *Circus* into CSP_M , with an implementation based on a published manual translation scheme. This scheme includes new and modified translation rules that emerged as a result of experimentation. We addressed issues with FDR state-space explosion, by optimising our models using the *Circus* Refinement Laws. We briefly describe the usage of *Circus2CSP* along with a discussion of some experiments comparing our tool with the literature.

1 Introduction

Among the range of verification techniques, model checking is used for exploring all the possible states a reactive system can reach. In other words, the algorithms used for model-checking will evaluate all possible scenarios of a system, which may eventually reach an undesired one, depending on the property that is being evaluated. Moreover, the focus of model-checking is on the system's behaviour rather than how the model would manage its data. Therefore, a system whose behaviour strongly relies on its data may become difficult to check, since the data may range over infinite domains.

There has been an effort from the community in order to design a systematic approach for model-checking *Circus*, which due to its combination of formalisms, is quite a challenge. *Circus* [34] is a formal language that combines structural aspects of a system using the Z language [36] and the behavioural aspects using CSP [32], along with the refinement calculus [22] and Dijkstra's guarded commands [7]. Its semantics is based on the *Unifying Theories of Programming (UTP)* [15]. As an initial attempt to model-check *Circus*, we participated in the ABZ'16 haemodialysis case study [12], producing a *Circus* specification, manually translating it into CSP_M , which we then checked with FDR[9]. Moreover, when translating *Circus* into CSP, we adapted the *Circus* model to map the structural Z parts into appropriate CSP.

Unlike in *Circus* processes, an explicit notion of state variables is not present in CSP processes. Therefore, in order to translate *Circus* state, we would either

translate it into a memory process [23,17,29], allowing other processes to read and write the values by synchronising on memory ‘get’ and ‘put’ events, or to transform the state variables into process parameters, as used by Beg [4]. For instance, we captured the state-based features of *Circus* in CSP using a memory process synchronising on channels for reading and updating the values of the state variables. Such an approach was also used while model checking [10] the ARINC 653 [2] architecture.

In this paper, we present *Circus2CSP*³ [13], a tool capable of model-checking specifications designed in *Circus* using FDR. It was developed by extending JAZA [33], a Z animator written in Haskell, in order to cover the *Circus* abstract syntax. The rest of the paper is organized as follows: In Sec. 2, we discuss the main goal of this work. A brief description of some experiments using *Circus2CSP* is presented in Sec. 3. The paper is concluded in Sec. 4.

2 *Circus2CSP*: Requirements and Goals

Our translation is based on that developed by Oliveira in the Compass project [26,27], which is based on repeated application of carefully selected *Circus* refinement laws, all of which happen to be equivalences. Such a translation uses set of rules for refining state-rich *Circus* into stateless processes that can be mapped into CSP_M .

Our focus while model-checking *Circus* is to produce a model in CSP_M where FDR can evaluate using as little computing resources as possible. As such, we provide a refined model from the strategy presented by Oliveira *et al.*[26], where our tool is capable of producing CSP_M models from larger specifications and making it possible for model-checking them using FDR. We highlight that because FDR is a refinement checker, it is not possible to perform temporal logic checks, which is further discussed by Lowe [20].

The entire toolset is developed as an extension of JAZA, which parses Z specifications written in L^AT_EX, the same input used by the Community Z Tools. Our goal was to produce a framework using the infrastructure available from JAZA, where the parser for Z was extended and now supports *Circus* and from there, we include new modules like the translation tool and the refinement calculator for *Circus*. Moreover, our tool is linked to FDR, and may also be integrated with other tools in the future. Our contribution here is mainly related to the fulfilment of a tool for automatically model-checking *Circus*.

The reason we adopted the translation presented by Oliveira *et al.*, is that, even though it is a manual translation, with no tool support involved, each translation step is justified by the *Circus* refinement laws, which have been formally proved to be correct. Currently, their approach covers a subset of *Circus*. However, our investigation [14] through experiments with the implementation of such rules demonstrated that such an initial and theoretical approach was restricted to a subset of the possible *Circus* specifications: those dealing with only one same

³ <https://bitbucket.org/circusmodelcheck/circus2csp>

type for all variables within the state of those processes. Thus, we had to implement not only a tool for the translation but also to refine that translation strategy in order to support a more realistic set of specifications: those using mixed types among their state variables.

We also experimented with the efficiency of FDR concerning the scale of the specifications. For such, we used the haemodialysis case study [12,3], a complex system which behaves according to the values of dozens of state variables. Thus, we refined the memory model in order to optimise the task of reading and updating the state variables from the *Circus* processes.

The outcome is that we now have a mechanised translator from *Circus* to CSP_M that produces tractable models, and allows the use of FDR on larger case studies than have been possible up to now. The new developed approach, as described in this paper, is sound since we were able to prove, by hand as well as using FDR as a refinement checker, that the memory model from Oliveira *et al.* is refined by the model discussed here [11, p. 77].

As part of the translation, we implemented an automatic refinement calculator for *Circus2CSP*, which handles a selected set of *Circus* refinement laws used according to [26, Appendix A, p. 147]. Moreover, we experimented with a strategy for refining Z schemas into "schema-free" *Circus* actions using Z Refinement Calculus [6].

Deliverables In summary, our research towards model checking *Circus* resulted in the following contributions:

- **A tool for automatically translating a subset of *Circus* into CSP_M :** Implementation of a tool based on the work of Oliveira *et al.* [26] where one is able to translate *Circus* models written in \LaTeX into CSP_M , and then, be able to perform model-checking and refinement checks using FDR.
- **An automatic *Circus* refinement calculator:** As part of the translation strategy, the *Circus* refinement laws are applied to the processes and actions. In order to automate the translation as much as possible, we provide an automatic *Circus* refinement calculator.
- **A transformation of some Z schemas into appropriate *Circus* constructs for translating into CSP_M :** The translation approach presented by Oliveira does not handle Z schemas directly, but only after normalisation. However, such a translation was not yet formally proved to be correct. We explore ways of translating Z schemas into *Circus* actions, specifically, those schemas where the translation results in a set of assignments.
- **An improved *Circus* model that supports multiple types within a specification:** The generated CSP_M model from Oliveira *et al.* using multiple types is not supported by FDR, since it contains some auxiliary functions that are seen by FDR as polymorphic functions, which are not supported by such a tool. We, however, introduce a new data structure that treats each type with its own set of auxiliary functions.
- **A refinement of the memory model from Oliveira *et al.* [26]:** We provide a refined memory model with distributed memory cells updating and

retrieving the values of the state variables, allowing FDR to handle a large number of state variables in a process, optimizing FDR’s effort to check such models.

- **New rules for mapping *Circus* to CSP_M :** We extended the mapping functions for expressions and predicates from Z , as well as mapping functions for those actions specifically related to the Memory model.
- **A mechanism that integrates *Circus2CSP* with FDR:** We connected our tool to the ”terminal-mode” interface of FDR, in order to be able to run checks straight from our tool. Unfortunately, we have no direct access to the code of FDR, and thus, we have to manually parse the results from the execution of FDR’s ”refine” command.
- **An automatic assertion generator for checking with FDR:** Our tool is able to generate assertion checks for refinement, deadlock, livelock and determinism checks for the loaded specification.

Tool Restrictions Our tool expects *Circus* specifications as input, written as a sequence of L^AT_EX block environments, very similar to the way Z paragraphs are written in L^AT_EX, which is a *de facto* standard for writing *Circus* specifications. However, we assume that the *Circus* document is already type checked by existing tools [21].

Our tool supports most of the *Circus* syntax, avoiding those constructs not handled in [26, p78] such as: no writing to input variables; external choice only among prefixed actions (those guaranteed to participate in an event before doing anything else, such as assignment); and no miraculous specifications.

Furthermore, some features are not yet supported such as: dealing with state invariants or preconditions in the Z schemas; non-determinism of data is not supported; and the consequences of nested parallelism and hiding with non-disjoint name sets have not been handled yet. These are a consequence of this being an automated translation, rather than the manual one prescribed in [26]. Finally, the translation of Z schemas used as *Circus* actions is restricted to those resulting in assignments.

3 Experiments with *Circus2CSP*

During our research we performed tests using our tool, *Circus2CSP*, exploring ways of overcoming any limitations from FDR, as well as comparing our approach with others from the literature.

Firstly, we explore the interference of invariants and preconditions in CSP_M , using the chronometer model from Oliveira [25, p.34-41], comparing the model from *Circus2CSP* with the translation from Oliveira [26]. We identified that using *Circus2CSP* the time spent by FDR to check for deadlock freedom, for example, with a model with the natural numbers ranging from zero to sixty (0..60), was of around 3 minutes. However, using Oliveira’s approach it took nearly three hours. In terms of states visited, FDR explored around 25 thousand states, compared with 99 million using the approach of Oliveira. In general, the CSP_M models

translated using our tool were evaluated by FDR using a much smaller state space and were checked in up to 95% less time than all the other models we tried derived from Oliveira's. However, we observed no correlation between time and state visited, in spite of the use (or not) of compression by default in FDR.

Then, we compare the translation of the haemodialysis case study using *Circus2CSP* with the *Circus* model from [12], which was manually translated into CSP_M for model-checking. We observed a reduction of over 91% of the state explored, as well as the execution time. Moreover, the manual translation didn't allow us to run FDR with a larger range of values for natural numbers, usually ranging from 0 up to 2. However, with *Circus2CSP*, we were able to go beyond the range 0 up to 90 in less than a minute. Such a result demonstrated that our approach is capable of handling large-scale case studies like the haemodialysis machine [12] and the ring buffer [38,26].

We also the effects of using some compression techniques available in FDR using the haemodialysis case study as an example. Although the states/transitions/plies visited were considerably reduced using the compression techniques such as *sbsim*, which determines the maximal strong bisimulation [5], and *wbsim*, which computes the maximal weak bisimulation, there was little impact on overall execution time, and the number of states visited are independent of the range of natural numbers used, while the number of transitions grows slowly. However, it is difficult to identify which compression technique will be most effective in a general case, and indeed, further experiments are required.

Finally, we compare different approaches for modeling the Ring Buffer case study [26,38], using FDR, in order to test the capabilities of our tool while model-checking the translated models, in contrast to the limitations of ProB [19]. Unfortunately, the structure defined for our translation strategy is not fully supported by ProB, which was used to test the model generated with the translation strategy from Ye [38]. ProB is another model-checker, which was originally developed for the B language, and was extended to support CSP, Z, Event-B [1], as well as combined languages such as CSP||B. We observed that some of the constructs used in our CSP_M model, such as `subtype`, are not yet supported by ProB. Nevertheless, we were able to use ProB's animator and to execute the same assertion check, as in FDR, obtaining similar results.

However, the tests performed with the CSP_M specification of Ye using FDR failed to check for deadlock freedom and determinism. The results obtained from ProB can be related to what we obtained in FDR in terms of the behavior of the system: the counterexample given from FDR can be used to animate the CSP||B model in ProB, causing the same effect: deadlock. Although, our experiment was limited since CSP||B takes into account the system state in ProB. In such model, the CSP_M file generated from Ye captures only the behavior of the system, but does not capture the system state. We reckon that the deadlock was caused because the state (modeled in B) can interfere in the system behavior in order to avoid deadlocks.

4 Conclusions

In this paper, we briefly introduced *Circus2CSP*, a tool capable of model-checking *Circus* specifications using FDR, through a translation strategy from *Circus* into CSP_M . It comprises a series of translation rules, combined with *Circus* refinement laws. One can perform refinement checks using FDR directly from *Circus2CSP*'s command-line. The tool can be downloaded freely from <https://bitbucket.org/circusmodelcheck/>.

We improved Oliveira's [26] translation strategy in a few ways: handling a wider mix of datatypes; translating Z schemas easily "compiled" to assignments; coping better with potentially large state spaces; and close integration with FDR. Some of the equivalence laws used in the translation have side-conditions that lead to proof obligations. Our tool does not discharge these, leaving them to the user to handle by other means.

The modifications for the memory model developed for our tool are similar to what was presented by Mota *et al.* [24], where interleaving between processes, one for each state variable, was proposed. In fact, the memory model used in [26] was based on the one by Mota *et al.*, and was expanded with the inclusion of a *terminate* signal, and, rather than one process for each variable, it would offer all possible *mget* and *mset* for all state variables at the same time.

However, after implementing Oliveira's memory model, we identified that such an approach tends to make the job of FDR harder and expensive, leading to state space explosion, when using several state variables on the same memory model. In our research, we also seek ways of optimising the CSP_M model in order to obtain a more efficient analysis. Therefore, even though the *terminate* event is used for synchronising the end of the *Memory* execution, the parallel composition of all possible *mgets* and *msets* leads to a possible exponential growth of the state space. However, such a problem does not occur while using interleaving. Moreover, Mota *et al.* also argues that the use of interleaving helps the compression algorithms built in FDR [30] to reduce the state space exploration while analysing such models.

A key principle in critical software development methods is that all global variables should be initialised pretty much immediately [2]. In a *Circus* context, if all the assignments are done before any observable event occurs, then its behaviour is that of a (simultaneous) assignment $s' = s_{init}$, where s is the (aggregated) global state. This allows us to introduce an additional translation step that replaces a non-deterministic choice over all possible starting values of s by one arbitrary choice of starting value for s . This is normally a proper refinement, but with initialisation as above, results in being an equivalence. This trick dramatically improved the performance of FDR.

Some related work on techniques for model-checking *Circus* was presented by Freitas [8] where a refinement model checker based on automata theory [16] and the operational semantics of *Circus* [35] was formalised in Z/Eves [31]. He also prototyped a model checker in Java. However, Freitas's *Circus* model checker is restricted to a subset of *Circus* actions and does not support the notion of *Circus* processes. Moreover, Nogueira *et al.* [23] also presented a prototype of a

model checker based on the operational semantics of *Circus* within the Microsoft FORMULA [18] framework. However, they could not provide a formal proof of the soundness of their approach, since FORMULA does not have an available formal semantics.

Yet another approach for model-checking *Circus* was defined by Ye and Woodcock [37], who defined a link from *Circus* to $CSP\|B$ with model-checking using ProB [28]. However, ProB is a limited tool in terms of processing capabilities: it does not support multiprocessors nor multithreading, and therefore, it is unlikely that it would support larger models such as those checked using *Circus2CSP*. Moreover, the translation proposed by Ye produces a specification split in two languages: B for the state machine and CSP for the behaviour of the model. However, such an approach is limited to be checked by ProB while ours isn't limited to FDR: we can indeed load our CSP_M translated models in ProB, and still perform model-checking. Finally, Beg [4] prototyped and investigated an automatic translation that supports a subset of *Circus* constructs, supporting only *Skip*, prefixing action, sequential composition, assignments, if statements, and guards with simple predicates.

For future work, we have plans for specifying a translation strategy for Z schemas used as *Circus* actions within a process. The best approach would be to use Z Refinement Calculus [6]. For now, our tool deals only with those schemas that in fact can be translated into assignments. We intend to explore the operators for Z schemas and the refinement laws that can be applied accordingly.

Acknowledgments This work was funded by CNPq (Brazilian National Council for Scientific and Technological Development) within the Science without Borders programme, Grant No. 201857/2014-6, and partially funded by Science Foundation Ireland grant 13/RC/2094.

References

1. Abrial, J.R.: Modeling in Event-B: System and Software Engineering. Cambridge University Press, New York, NY, USA, 1st edn. (2010)
2. Aeronautical Radio, I.A.: ARINC 653: Avionics Application Standard Software Interface (nov 2006)
3. Atif, M.: The hemodialysis machine case study. In: Butler, M.J., Schewe, K.D., Mashkoo, A., Biró, M. (eds.) Lecture Notes in Computer Science (including sub-series Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Lecture Notes in Computer Science, vol. 9675, pp. 329–343. Springer (2016), https://doi.org/10.1007/978-3-319-33600-8_29
4. Beg, A., Butterfield, A.: Development of a prototype translator from Circus to CSPm. In: ICOSST 2015 - 2015 International Conference on Open Source Systems and Technologies, Proceedings. pp. 16–23 (dec 2016)
5. Boulgakov, A., Gibson-Robinson, T., Roscoe, A.W.: Computing maximal weak and other bisimulations. Formal Aspects of Computing 28(3), 381–407 (2016), <https://doi.org/10.1007/s00165-016-0366-2>

6. Cavalcanti, A., Woodcock, J.C.P.: ZRC - A Refinement Calculus for Z. *Formal Aspects of Computing* 10(3), 267–289 (mar 1998), <http://link.springer.com/10.1007/s001650050016><https://doi.org/10.1007/s001650050016>
7. Dijkstra, E.W.: Guarded commands, nondeterminacy and formal derivation of programs. *Communications of the ACM* 18(8), 453–457 (1975), <http://portal.acm.org/citation.cfm?doid=360933.360975>
8. Freitas, L.: Model checking Circus. Ph.D. thesis, Department of Computer Science, The University of York, UK (2005)
9. Gibson-Robinson, T., Armstrong, P., Boulgakov, A., Roscoe, A.W.: FDR3 — A Modern Model Checker for CSP. *Tools and Algorithms for the Construction and Analysis of Systems* 8413, 187–201 (2014), <http://www.cs.ox.ac.uk/projects/fdr/manual/>
10. Gomes, A.O.: Formal Specification of the ARINC 653 Architecture Using Circus (2012), <http://theses.whiterose.ac.uk/id/eprint/2683>
11. Gomes, A.O.: Model-Checking Circus with FDR using Circus2CSP. Ph.D. thesis, Trinity College Dublin (2019), <http://www.tara.tcd.ie/handle/2262/86009>
12. Gomes, A.O., Butterfield, A.: Modelling the haemodialysis machine with Circus. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 9675(201857), 409–424 (2016), https://doi.org/10.1007/978-3-319-33600-8_34
13. Gomes, A.O., Butterfield, A.: Circus2CSP - A Translator from Circus to CSPm (2018), <https://bitbucket.org/circusmodelcheck/circus2csp>
14. Gomes, A.O., Butterfield, A.: Towards a Model-Checker for Circus. In: *In This Volume*. Springer Berlin Heidelberg, Berlin, Heidelberg (2019)
15. Hoare, C., He, J.: *Unifying Theories of Programming*. Prentice-Hall (1998)
16. Hopcroft, J.E., Motwani, R., Ullman, J.D.: *Introduction to automata theory, languages, and computation - international edition (2. ed)*. Addison-Wesley (2003)
17. Hopkins, D., Roscoe, A.W.: SVA , a tool for analysing shared-variable programs. *Electronic Notes in Theoretical Computer Science* pp. 1–5 (2007), <http://www.cs.ox.ac.uk/people/bill.roscoe/publications/119.pdf>
18. Jackson, E.K., Levendovszky, T.T., Balasubramanian, D.: Reasoning about meta-modeling with formal specifications and automatic proofs. In: Whittle, J., Clark, T., Kühne, T. (eds.) *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. *Lecture Notes in Computer Science*, vol. 6981 LNCS, pp. 653–667. Springer (oct 2011), https://doi.org/10.1007/978-3-642-24485-8_48
19. Leuschel, M., Butler, M.: ProB: An automated analysis toolset for the B method. *International Journal on Software Tools for Technology Transfer* 10(2), 185–203 (2008)
20. Lowe, G.: Specification of communicating processes: Temporal logic versus refusals-based refinement. *Formal Aspects of Computing* 20(3), 277–294 (2008), <https://link.springer.com/content/pdf/10.1007/s00165-007-0065-0.pdf>
21. Malik, P., Utting, M.: CZT: A Framework for Z Tools. *ZB 2005: Formal Specification and Development in Z and B* 3455, 315–352 (2005), <http://czt.sourceforge.net>https://doi.org/10.1007/11415787_5
<https://dx.doi.org/10.1007/11415787-5>
22. Morgan, C., Carroll: *Programming from specifications (2nd. Ed.)*, Prentice Hall International series in computer science, vol. 16. Prentice Hall (1994), <https://dl.acm.org/citation.cfm?id=184737>

23. Mota, A., Farias, A.C., Didier, A., Woodcock, J.C.P.: Rapid prototyping of a semantically well founded Circus model checker. In: Giannakopoulou, D., Salaün, G. (eds.) *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Lecture Notes in Computer Science, vol. 8702 LNCS, pp. 235–249. Springer (2014), https://doi.org/10.1007/978-3-319-10431-7_17
24. Nogueira, S., Sampaio, A., Mota, A.: Test generation from state based use case models. *Formal Aspects of Computing* 26(3), 441–490 (2014)
25. Oliveira, M.V.M.: *Formal Derivation of State-Rich Reactive Programs using Circus*. Ph.D. thesis, University of York, UK (2005), <http://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.428459>
26. Oliveira, M.V.M., Sampaio, A., Antonino, P., Ramos, R., Cavalcanti, A., Woodcock, J.C.P.: *Compositional Analysis and Design of CML Models*. Tech. Rep. D24.1, COMPASS Deliverable (2013), <http://www.compass-research.eu/Project/Deliverables/D241.pdf>
27. Oliveira, M.V.M., Sampaio, A., Conserva Filho, M.: Model-Checking Circus State-Rich Specifications. *Integrated Formal Methods 2014* 8739 LNCS, 39–54 (2014), https://doi.org/10.1007/978-3-319-10181-1_3
28. Plagge, D., Leuschel, M.: Validating Z Specifications Using the ProB Animator and Model Checker. In: Davies, J., Gibbons, J. (eds.) *Integrated Formal Methods*, *Lecture Notes in Computer Science*, vol. 4591, pp. 480–500. Springer (2007), https://doi.org/10.1007/978-3-540-73210-5_25
29. Roscoe, A.W.: *The Theory and Practice of Concurrency*. Prentice Hall PTR, Upper Saddle River, NJ, USA (1973)
30. Roscoe, A.W.: *Understanding Concurrent Systems*, *Texts in Computer Science*, vol. 42. Springer London, London (2010), <http://www.ncbi.nlm.nih.gov/pubmed/20549881>
31. Saaltink, M., Meisels, I., Saaltink, M.: *The Z/EVES Reference Manual (for version 1.5)*. Reference manual, ORA Canada pp. 72–85 (1997), <http://dl.acm.org/citation.cfm?id=647282.722913>
32. Schneider, S.: *Concurrent and Real-time systems*. Wiley Chichester, UK (2000)
33. Utting, M.: *Jaza User Manual and Tutorial* (jun 2005)
34. Woodcock, J.C.P., Cavalcanti, A.: The Semantics of Circus. In: *Zb 2002: Formal Specification and Development in Z and B: 2nd International Conference of B and Z Users* Grenoble (2002)
35. Woodcock, J.C.P., Cavalcanti, A.L.C., Freitas, L.: Operational Semantics for model checking Circus. *Lecture Notes in Computer Science* 3582, 237–252 (2005)
36. Woodcock, J.C.P., Davies, J.: *Using Z, Specification, Refinement, and Proof*. Prentice Hall International Series in Computer Science, Prentice-Hall, Inc., Upper Saddle River, NJ, USA (1996)
37. Ye, K.: *Model Checking of State-Rich Formalisms*. Ph.D. thesis, University of York (2016)
38. Ye, K., Woodcock, J.C.P.: Model checking of state-rich formalism Circus by linking to CSP—B. *International Journal on Software Tools for Technology Transfer* 19(1), 73–96 (2017), <https://doi.org/10.1007/s10009-015-0402-1>