Refinement in Integrated Specifications of CSP-OZ

Azman Bujang Masli, Edwin Mit , Nurfauza Jali and Yanti Rosmunie Bujang
Faculty of Computer Science and Information Technology
Universiti Malaysia Sarawak
Kota Samarahan, Sarawak, Malaysia
{bmazman, edwin, jnurfauza, byanti}@fit.unimas.my

***Abstract*—Formal specification provides the means to verify a system's correctness and this can be done by the development technique of *refinement* of formal specification. Considering the multiple views of a system, in integrated formal specifications, will introduce more than one refinement that can be applied to the specification. This paper investigates the interaction of the different parts of an integrated specification under refinement. The integrated notation of CSP-OZ integrates the behaviour based language CSP with the state based notation, Object-Z. In such integrated notation, not only different views of a system are available, but the refinement relations in both parts are also of different basis.**

I. Introduction

First introduced by Dijkstra [1] and Wirth [2] in the early seventies, *refinement* is a step-by-step process of development where a specification is transformed into more detail or concrete specification. The idea behind refinement is the notion of substitutivity where a specification can be replaced with another specification without noticeable behaviour being detected by users [3]. The process involves resolving uncertainties in the abstract specification, as well as making it more implementable in the target programming language. The uncertainties of the abstract specification could be in terms of choices that are left open or underspecified operations.

A number of work have been done on combining the different views of formal specification (see, e.g., [4], [5], [6]). The idea behind all approaches to such a combination is to define a comparable semantics [7] for the different specification languages, mostly by interpreting one language into the semantics of the other. Consequently this means that, if we have an integration of two different specification languages, we can apply at least two different refinements to the specification as well. This paper discusses the application of refinement of different languages in such setting. Specifically we will look into the problems that might occur in one part of an integrated specification when we apply refinement to the other part of it. The discussion will be based around the integrated specification language of Communicating Sequential Processes (CSP) [8] and Object-Z [9], called CSP-OZ, as defined in [10]. [11] and [12] also integrate CSP with Object-Z using simpler approaches.

II. Related Work

State-based view of communicating processes has been explored in [13] and [14] where simulation techniques for processes, similar to those found in Z, are developed and proved. The approach in [13] employs labelled state-transition systems in deriving corresponding CSP semantics for processes and their refinements in a state-based setting. Another similar work has also been done in [15] by adopting weakest preconditions over action systems, and furthermore [16] (also in [17]) investigated the same problem by deriving concurrent refinement in relational setting.

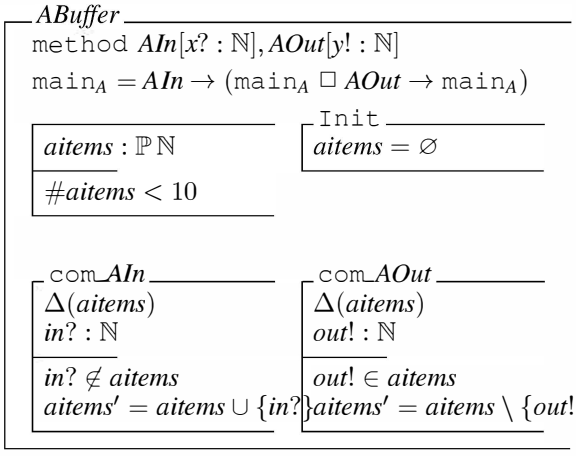
In integrated specifications, refinement in one view is applicable by assuming the other view of the specification remains unchanged. This is the approach taken in [18] to define data refinement for CSP-CASL specification. The same approach is also used in [10], which defines data refinement for CSP-OZ specification.

This paper presents and investigates the refinements of CSP-OZ specification, where more than one refinement relations exist and can be applied by restricting the changes in the other part of the specification.

III. The CSP-OZ

The combination of CSP and Object-Z discussed in this paper, is based on the one described in [19], which is called CSP-OZ, where Object-Z classes are also given a CSP semantics. That is, the syntax of an Object-Z class has been extended to include the definition of the CSP process of the class and channels, which is called the CSP part of it.

The following is an example of a CSP-OZ specification of a buffer. Due to some conventions used in CSP-OZ specification, we add keyword "com" to the operation names to reflect the fact that we use the blocking mode of operations. The CSP part of the class defines the behaviour or the sequence of operations taking place.



In the class *ABuffer* above, the CSP part of the class offers a choice to the environment after event *AIn* has taken place.

978-1-4673-6722-6/15/$31.00 ©2015 IEEE

Authorized licensed use limited to: UNIVERSITY MALAYSIA SARAWAK. Downloaded on September 15,2022 at 01:35:14 UTC from IEEE Xplore. Restrictions apply.

The subsequent event could be either *Ain* or *AOut* and this is determined by the user or environment of the *ABuffer* process.

IV. Refinement in CSP-OZ

The specification of a CSP-OZ class comprises of two parts, where the CSP part defines the behaviour of the class as a CSP process and the OZ part defines the class' data structure and operations.

Although the OZ part is given CSP semantics and thus can be treated as a CSP process by itself, however, Z refinement simulations are still needed in order to refine the class' operations and data types. This is because there is no state information available in the semantic representation, either from the operational or the denotational semantics, of a CSP process [10, pp 106]. Therefore, Z refinement is used in order to deal with the state information in the OZ part of CSP-OZ specification.

The *failures* semantics of CSP only records the availability of events that correspond to operations of the OZ part, and not the resulting state of the class after an operation is executed. For example the following class *A* may end in either one of two different states, in the OZ part of it, and this is due to the non-determinism in operation Op_1. This information, however, is not captured in the process of the CSP part. Thus, in order to deal with the non-determinism in the OZ part of CSP-OZ specifications, Z refinement can be applied in the OZ part, while CSP refinements are applicable to the process in the CSP part.

A

$\texttt{method}\ Op_1, Op_2$

$\texttt{main}_a = Op_1 \rightarrow \textsc{Stop} \,\Box\, Op_1 \rightarrow Op_2 \rightarrow \texttt{main}_a$

	Init	com_Op_1	com_Op_2
$x_a : \mathbb{N}$	$x_a = 0$	$\Delta(x_a)$	$\Delta(x_a)$
		$x'_a \in \{1, 2\}$	$x_a = 1 \Rightarrow x'_a = 3$
			$x_a = 2 \Rightarrow x'_a = 4$

Therefore, given a CSP-OZ class $CSP_A \| OZ_A$, and assuming that the CSP part remains unchanged, one can apply Z refinement to it to become a more concrete class [10], lets say $CSP_B \| OZ_B$, with a retrieve relation R. We can say that $CSP_A \| OZ_A \sqsubseteq_{\mathcal{Z}} CSP_B \| OZ_B$ under retrieve relation R with $CSP_A = CSP_B$. Such refinement is provable to hold by either forward or backward simulation of Z refinement or both. For example we can reduce the non-determinism in class *A* by refining the OZ part as follows

C

$\texttt{method}\ Op_1, Op_2$

$\texttt{main}_c = Op_1 \rightarrow \textsc{Stop} \,\Box\, Op_1 \rightarrow Op_2 \rightarrow \texttt{main}_c$

	Init	com_Op_1	com_Op_2
$x_c : \mathbb{N}$	$x_c = 0$	$\Delta(x_c)$	$\Delta(x_c)$
		$x'_c = 1$	$x_c = 1 \Rightarrow x'_c = 3$
			$x_c = 2 \Rightarrow x'_c = 4$

which is a forward simulation of Z refinement with the following retrieve relation *R*

R

$A.\textsc{State}$

$C.\textsc{State}$

$x_a \in \{1, 2\} \Leftrightarrow x_c = 1$

The same technique is also applicable when we consider CSP refinement for the CSP part of a CSP-OZ class. Assuming the OZ part remains the same, either *traces* or *failures* refinement can be applied to the CSP part of the class and the resulting concrete class is a refinement of the abstract one under the chosen refinement. For example, we apply traces refinement to the CSP part of class *C* above as follows

D

$\texttt{method}\ Op_1, Op_2$

$\texttt{main}_d = Op_1 \rightarrow Op_2 \rightarrow \texttt{main}_d$

	Init	com_Op_1	com_Op_2
$x_d : \mathbb{N}$	$x_d = 0$	$\Delta(x_d)$	$\Delta(x_d)$
		$x'_d = 1$	$x_d = 1 \Rightarrow x'_d = 3$
			$x_d = 2 \Rightarrow x'_d = 4$

This, however, is not true if there are changes made to the other part of the class while refining the other, where the changes result in incompatibility (that is not a refinement). For example, the following class *E* is not a refinement of class *C* due to the changes made in the OZ part of the class with an extra operation (Op_3) has been added.

E

$\texttt{method}\ Op_1, Op_2$

$\texttt{main}_e = Op_1 \rightarrow Op_2 \rightarrow \texttt{main}_e$

	Init	com_Op_1	com_Op_2
$x_e : \mathbb{N}$	$x_e = 0$	$\Delta(x_e)$	$\Delta(x_e)$
$y : \mathbb{P}\,\mathbb{N}$	$y = \varnothing$	$x'_e = 1$	$x_e = 1 \Rightarrow x'_e = 3$
		$y' = y \cup \{x_e\}$	$x_e = 2 \Rightarrow x'_e = 4$
			$y = y' \cup \{x_e\}$

Authorized licensed use limited to: UNIVERSITY MALAYSIA SARAWAK. Downloaded on September 15,2022 at 01:35:14 UTC from IEEE Xplore. Restrictions apply.

com_Op_3

$\Delta(x_e)$

$x'_e = x + 1$
$y' = y \cup \{x_e\}$

This suggests that both refinements, of the CSP part and the OZ part, may be done at the same time and the resulting class will still be a valid refinement of the abstract one. For example, class D is also a refinement of class A, where both parts have been refined.

As we can observe from the example, we have both traces and Z refinements between classes A and D. That is given $A = CSP_A \| OZ_A$ and $D = CSP_D \| OZ_D$, we have $CSP_A \sqsubseteq_{\mathcal{T}} CSP_D$ and $OZ_A \sqsubseteq_{\mathcal{Z}} OZ_D$. Therefore, we do not need to have equal OZ parts in order to refine the CSP part, or the other way round, as mentioned in [10]. All we need is that there exists a refinement[1] in one part while refining the other.

Another interesting observation that we can get from the example is that, we cannot apply the transitive property of refinement between classes A, C and D to imply that $A \sqsubseteq D$. Although we have a refinement between A and C as well as between C and D, these refinements, however, are of different ordering. The refinement that we have between A and C is a traces refinement where $A \sqsubseteq_{\mathcal{T}} C$ with $CSP_A \sqsubseteq CSP_C$, while Z refinement exists between C and D with $OZ_C \sqsubseteq OZ_D$. Thus the transitive property of refinement $(A \sqsubseteq C) \wedge (C \sqsubseteq D) \Rightarrow A \sqsubseteq D$ does not work in this case. The transitivity of refinement in this case can be worked out at the semantic level by showing that the semantic of D is within the semantic of A. We will not pursue this here at the moment, but will consider it as part of future works.

A. The CSP Part and Z Refinement

Whenever there is non-determinism in an operation of the OZ part, then we would expect that the operation may end up in more than one after state. Depending on how the specification is specified, each of the after state may lead to a different operation being enabled. Hence what operation follows next is determined by which state will become available.

As an example, consider the following CSP-OZ class P

P

method Op_a; Op_b; Op_c

main $= Op_a \rightarrow Op_b \rightarrow STOP \;\Box\; Op_a \rightarrow Op_c \rightarrow STOP$

$x : \mathbb{N}$

Init

$x = 0$

com_Op_a

$\Delta(x)$

$x = 0$
$x' \in \{1, 2\}$

com_Op_b

$\Delta(x)$

$x = 1 \Rightarrow x' = 3$

com_Op_c

$\Delta(x)$

$x = 2 \Rightarrow x' = 4$

where the non-determinism in Op_a results in either Op_b or Op_c will be enabled. Here we have two different sequences of operations in the OZ part of class P and the non-determinism is reflected in the CSP part as well, where event Op_a leads to two different events.

On the other hand, for the second scenario, the non-determinism in the operation of the OZ part may be handled by only one operation as shown in the following CSP-OZ class Q.

Q

method Op_a; Op_b

main $= Op_a \rightarrow Op_b \rightarrow STOP$

$x : \mathbb{N}$

Init

$x = 0$

com_Op_a

$\Delta(x)$

$x = 0$
$x' \in \{1, 2\}$

com_Op_b

$\Delta(x)$

$x = 1 \Rightarrow x' = 3$
$x = 2 \Rightarrow x' = 4$

In this case, there is only one possible sequence of operations. Thus no non-determinism, due to the non-determinism in operation Op_a of the OZ part, can be introduced in the process definition of the CSP part. Also, the trace in the CSP part will represent the sequence of operations of the OZ part that will give two possible paths.

Although the CSP part and the OZ part of a CSP-OZ specification represent two different CSP processes that interact via CSP parallel operator, since the process of the CSP part is usually defined within the behaviour of the OZ part, it is assumed that the following holds for every CSP-OZ specification within this paper.

$$traces(CSP) \subseteq traces(OZ)$$

This is also due to the fact that the CSP part restricts the order of operations of the OZ part. With this assumption in place, it is only in the first scenario where Z refinement in the OZ part of a CSP-OZ class could affect the process definition of the CSP part.

Thus resolving non-determinism of an operation by refining the OZ part will remove the corresponding non-determinism in the CSP part as well. This is because the containment of the traces should still hold in the concrete specification. This is shown in figure 1 below.

On the other hand, if there exists non-determinism in operations of the OZ part of a CSP-OZ specification that is not reflected in the process definition of the CSP part[2] as discussed

[1] This may also mean that the other part is equal as proposed in [10].

[2] There is no unique process specification for a CSP-OZ class, since we can define the order of the operations in any order. See [20], [21] for translating Z specification to CSP.

Authorized licensed use limited to: UNIVERSITY MALAYSIA SARAWAK. Downloaded on September 15,2022 at 01:35:14 UTC from IEEE Xplore. Restrictions apply.

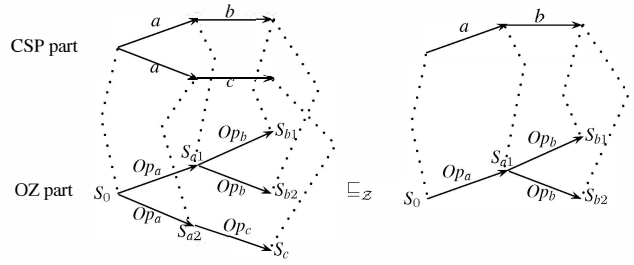


Fig. 1. Z refinement and the traces of the CSP part

in the second scenario, refining the OZ part to reduce the non-determinism will not affect the CSP part of a CSP-OZ specification. This is illustrated in figure 2 below, where we remove the non-determinism in Op_b. The restriction on traces

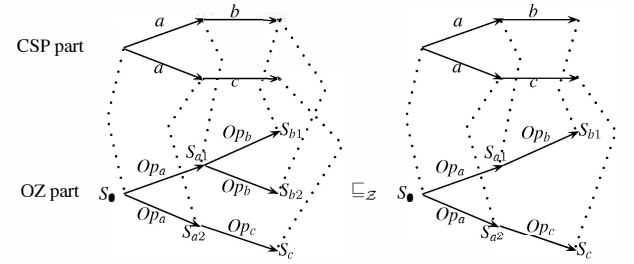


Fig. 2. Effects of refinement in the OZ part to the CSP part

containment, $traces(CSP) \subseteq traces(OZ)$, remains valid after the refinement.

B. The OZ Part and CSP Refinement

Since the process of the CSP part imposes restrictions on the execution of operations of the OZ part, applying CSP refinement to the CSP part could further escalate the restrictions. This is because the refinement in the CSP part will further reduce the possible traces of the specification. For example in traces refinement, traces of the concrete process should be contained in the traces of the abstract process. Failures refinement also requires traces containment between the traces of the concrete and abstract specifications. Thus, refining the process in the CSP part will further reduce the possible traces of the specification and these are the traces within which the operations of the OZ part will be executed.

Although the refinement does not directly change the specification of the OZ part, however, it does affect the operations that will become available and their order of execution. This affects the OZ part in such a way that some states may become unavailable. This subsection discusses on how traces and failures refinements of the CSP part affect the OZ part of a CSP-OZ specification.

1) CSP Traces Refinement: When traces refinement is applied to the CSP part, some (or part) of the traces might be removed. As an example consider the following CSP-OZ specification

A

$\texttt{method}\ Op_1, Op_2, Op_3$

$\texttt{main} = Op_1 \rightarrow Op_2 \rightarrow Op_3 \rightarrow STOP$

Init

$x : \{0, ..., 9\}$ | $x = 0$

com_Op_1

$\Delta(x)$

$x = 0$
$x' \in \{1, 2\}$

com_Op_2

$\Delta(x)$

$x = 1 \Rightarrow x' \in \{3, 4\}$
$x = 2 \Rightarrow x' = 5$

com_Op_3

$\Delta(x)$

$x = 3 \Rightarrow x' = 6$
$x = 4 \Rightarrow x' \in \{7, 8\}$
$x = 5 \Rightarrow x' = 9$

where traces refinement is applied to the CSP part to get the following process

$$\texttt{main} = Op_1 \rightarrow Op_2 \rightarrow STOP$$

As we can observe, the set of traces for the abstract CSP process is as follows

$$\{\langle\rangle, \langle Op_1\rangle, \langle Op_1, Op_2\rangle, \langle Op_1, Op_2, Op_3\rangle\}$$

and for the concrete process is

$$\{\langle\rangle, \langle Op_1\rangle, \langle Op_1, Op_2\rangle\}$$

Thus we have traces refinement in the CSP part. The refinement and the corresponding operations and states of the OZ part is shown in figure 3(A and B)[3]. The corresponding states of the OZ part are $S_0 = \{(x \rightsquigarrow 0)\}$, $S_{1a} = \{(x \rightsquigarrow 1)\}$, $S_{1b} = \{(x \rightsquigarrow 2)\}$, $S_{2a} = \{(x \rightsquigarrow 3)\}$, and so on. Due to the non-determinism in Op_1, the state of the specification after the operation is either in S_{1a} or S_{1b} as shown in figure 3. There are also non-determinism in the other operations as well.

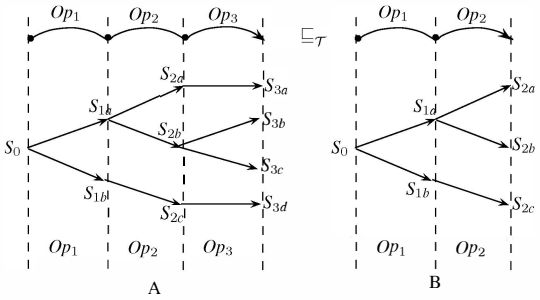


Fig. 3. Traces refinement and OZ part operations

After traces refinement is applied to the CSP part, operation Op_3 of the OZ part is no longer available in the concrete specification and the states S_{3a}, S_{3b}, S_{3c} and S_{3d} have also been removed (see figure 3B).

[3]Figure A for the abstract specification and B for the concrete specification.

Authorized licensed use limited to: UNIVERSITY MALAYSIA SARAWAK. Downloaded on September 15,2022 at 01:35:14 UTC from IEEE Xplore. Restrictions apply.

2) CSP Failures Refinement: Considering the failures refinement in the CSP part, the only impact that we may have to the OZ part is in terms of the way the non-determinism is resolved under the refinement. This is because the non-determinism that we have in the CSP part of a CSP-OZ specification are the consequence of the non-determinism that exist in the operations of the OZ part[4], removing one of them under failures refinement will disallow the corresponding non-determinism in the OZ part. This is illustrated in figure 4 below, where we refine the CSP part by removing the non-determinism in the initial event of the CSP part. The

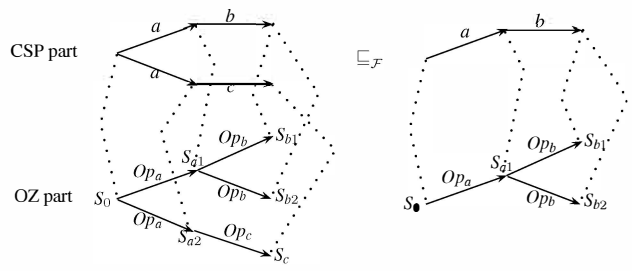
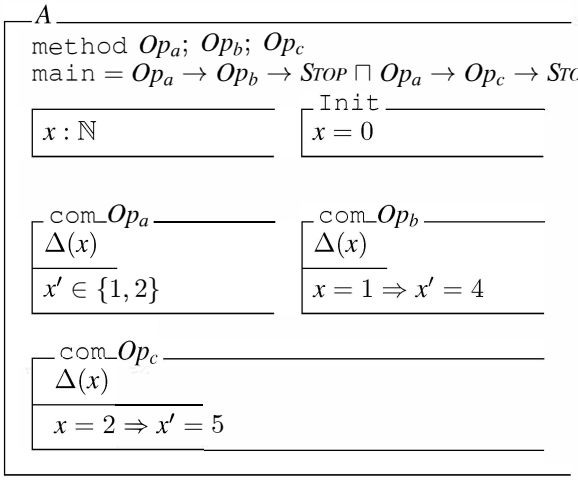


Fig. 4. Failures refinement affects the OZ part transitions

corresponding OZ part, as well as the concrete CSP part after the refinement, is shown in the right-hand side of the figure. We also remove the alternate path of the OZ part to avoid a deadlock.

As an example consider the following CSP-OZ specification of class A



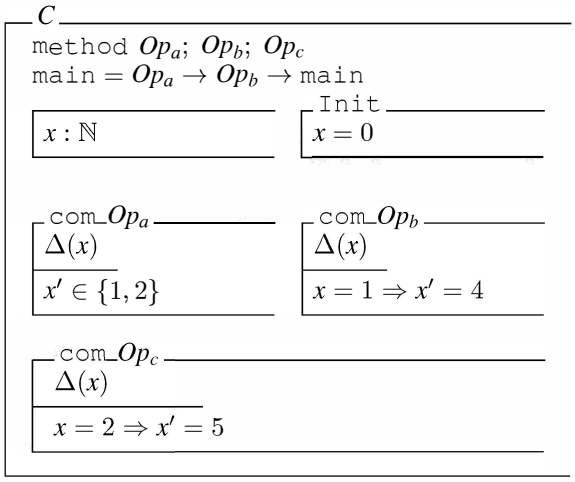
where we have non-determinism in Op_a, which non-deterministically assigns the value 1 or 2 to variable x. The non-determinism is reflected in the CSP part where both of the following set of traces are possible.

$$tr_1 = \{\langle\,\rangle, \langle Op_a\rangle, \langle Op_a, Op_b\rangle\}$$
$$tr_2 = \{\langle\,\rangle, \langle Op_a\rangle, \langle Op_a, Op_c\rangle\}$$

Based on the state of the OZ part after Op_a is executed, operation Op_b will be executed next if $x = 1$ and Op_c if $x = 2$.

[4]This is because $traces(CSP||OZ) \subseteq traces(OZ)$ holds. Thus for any non-determinism in $traces(CSP||OZ)$, there must exist the corresponding non-determinism in the OZ part, which is in the operation itself.

We apply failures refinement to the CSP part in order to remove the non-determinism in it, where the set of traces tr_2 is no longer possible, and keep the OZ part as it is in the following concrete class C.



Failures of A is the following

$$failures(A) = \{(\langle\,\rangle, \{Op_b, Op_c\}), (\langle Op_a\rangle, \{Op_a, Op_b\}), (\langle Op_a\rangle, \{Op_a, Op_c\}), (\langle Op_a, Op_b\rangle, \{Op_a, Op_b, Op_c\}), (\langle Op_a, Op_c\rangle, \{Op_a, Op_b, Op_c\})\}$$

while failures of C is as follows

$$failures(C) = \{(\langle\,\rangle, \{Op_b, Op_c\}), (\langle Op_a\rangle, \{Op_a, Op_c\}), (\langle Op_a, Op_b\rangle, \{Op_a, Op_b, Op_c\})\}$$

where $traces(C) \subseteq traces(A)$ and $failures(C) \subseteq failures(A)$, and therefore we have failures refinement with $A \sqsubseteq_{\mathcal{F}} C$.

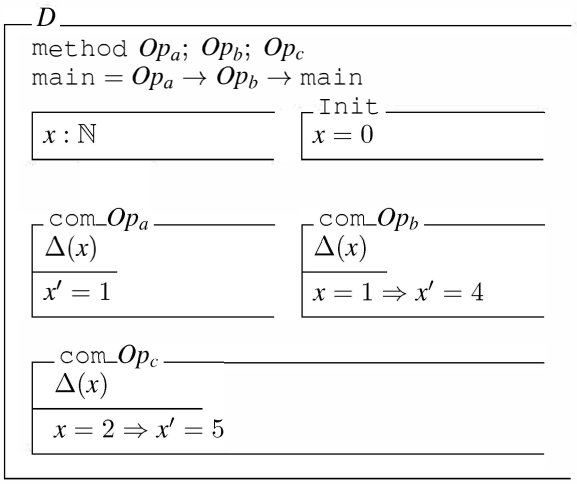
As we can observe, although the non-determinism in the CSP part has been resolved, the corresponding non-determinism in the OZ part remains. Since the CSP part defines the order of the operations to be executed, after the execution of operation Op_a, the OZ part is restricted to only continue if the state of the class is in the state where operation Op_b is possible. However, it is still possible for the OZ part to end up with the state where only operation Op_c can be executed and not Op_b. This is due to the fact that the CSP part only defines the order of the operations of the OZ part and has nothing to do with what is going on in them.

Therefore, removing non-determinism in the CSP part does not affect the way an operation in the OZ part is carried out. It does, however, affect in terms of what operation will be applicable next and this should confirm with the way the non-determinism in the CSP part has been resolved. In the case of the above example, only Op_b is allowed to execute after Op_a. Otherwise deadlock will occur where both parts cannot synchronise on the same event. This happens when $x = 2$ due to the non-determinism in Op_a. Note that, the assumption $traces(CSP) \subseteq traces(OZ)$ still holds after the refinement.

In order to avoid the deadlocking problem, we must further refine the OZ part to remove the non-determinism in it. Thus, although class C is a valid refinement of class A according to [10], by keeping the OZ part equal, it is however not acceptable to get a concrete class that may deadlock. Further refinement

Authorized licensed use limited to: UNIVERSITY MALAYSIA SARAWAK. Downloaded on September 15,2022 at 01:35:14 UTC from IEEE Xplore. Restrictions apply.

is needed in the OZ part to remove the problem before another refinement is applied to either part of the specification. In the case of the above example, we have to refine the OZ part as follows to avoid the deadlocking problem.



In class D above, we remove the non-determinism in Op_a, and as a result Op_c is no longer possible. We still have $A \sqsubseteq D$ hold with $CSP_A \sqsubseteq_{\mathcal{F}} CSP_D$ in the CSP part as well as $OZ_A \sqsubseteq_{\mathcal{Z}} OZ_D$ in the OZ part.

However, in the case where the non-determinism results in the same operation of the OZ part being executed, as discussed in the second scenario in subsection IV-A, no deadlock will occur and no further refinement is needed in the OZ part as well.

V. Conclusions

Refinement as a development technique in formal specification is very much related to the semantics adopted for the specification language. The integration of more than one specification languages, by intepreting the semantics of one into the other, does not mean that the refinement of the first is also applicable to the other. This is due to the fact that each refinement is defined based on the semantics adopted and different semantics interprets a specification at different level of detail. Hence, understanding the restrictions and influences of one refinement to the other part of an integrated specification can give insight into the potentials and limitations of a specification towards an implementation.

The integration of behavioural and state views of CSP-OZ has restricts the refinements that are applicable to either side of a CSP-OZ specification. This may as well affects the initial requirements of a specification (properties), where only certain requirements are preserved in the concrete specification. The results discussed in this paper can be used as the first step in verifying the fulfillment of requirements in refining an integrated specification.

Acknowledgements

The authors would like to acknowledge Universiti Malaysia Sarawak (UNIMAS) and Kementerian Pendidikan Malaysia for the supports and funding provided under the fudamental research grant no:FRGS/ICT07(01)/1071/2013 (17).

References

[1] E. W. Dijkstra, "Notes on structured programming," pp. 1–82, 1972.

[2] N. Wirth, "Program development by stepwise refinement," *Commun. ACM*, vol. 14, no. 4, pp. 221–227, 1971.

[3] J.-R. Abrial, *The B-book: assigning programs to meanings*. New York, NY, USA: Cambridge University Press, 1996.

[4] J. C. P. Woodcock and A. L. C. Cavalcanti, "A concurrent language for refinement," in *IWFM'01: 5th Irish Workshop in Formal Methods*, ser. BCS Electronic Workshops in Computing, A. Butterfield and C. Pahl, Eds., Dublin, Ireland, July 2001.

[5] C. Bolton and J. Davies, "Refinement in Object-Z and CSP," in *IFM*, ser. Lecture Notes in Computer Science, M. J. Butler, L. Petre, and K. Sere, Eds., vol. 2335. Springer, 2002, pp. 225–244.

[6] J. Derrick, "Timed CSP and Object-Z," in *ZB 2003: Formal Specification and Development in Z and B*, ser. Lecture Notes in Computer Science, D. Bert, J. Bowen, S. King, and M. Walden, Eds., vol. 2651. Springer, June 2003, pp. 300–318. [Online]. Available: http://www.cs.kent.ac.uk/pubs/2003/1626

[7] C. Bolton and J. Davies, "A comparison of refinement orderings and their associated simulation rules," *Electr. Notes Theor. Comput. Sci.*, vol. 70, no. 3, 2002.

[8] S. D. Brookes, C. A. R. Hoare, and A. W. Roscoe, "A theory of Communicating Sequential Processes," *J. ACM*, vol. 31, no. 3, pp. 560–599, 1984.

[9] G. Smith, *The Object-Z specification language*. Norwell, MA, USA: Kluwer Academic Publishers, 2000.

[10] C. Fischer, "Combination and implementation of processes and data: from CSP-OZ to Java," Ph.D. dissertation, University of Oldenburg, January 2000. [Online]. Available: http://csd.Informatik.Uni-Oldenburg.DE/pub/Papers/fischer/cspoz2java.ps.gz

[11] G. Smith, "A semantic integration of Object-Z and CSP for the specification of concurrent systems," in *FME*, ser. Lecture Notes in Computer Science, J. S. Fitzgerald, C. B. Jones, and P. Lucas, Eds., vol. 1313. Springer, 1997, pp. 62–81.

[12] G. Smith and J. Derrick, "Abstract specification in Object-Z and CSP," in *ICFEM*, ser. Lecture Notes in Computer Science, C. George and H. Miao, Eds., vol. 2495. Springer, 2002, pp. 108–119.

[13] M. B. Josephs, "A state-based approach to communicating processes," *Distributed Computing*, vol. V3, no. 1, pp. 9–18, March 1988. [Online]. Available: http://dx.doi.org/10.1007/BF01788563

[14] J. He, "Process simulation and refinement," *Formal Asp. Comput.*, vol. 1, no. 3, pp. 229–241, 1989.

[15] J. Woodcock and C. Morgan, "Refinement of state-based concurrent systems," in *VDM Europe*, ser. Lecture Notes in Computer Science, D. Bjørner, C. A. R. Hoare, and H. Langmaack, Eds., vol. 428. Springer, 1990, pp. 340–351.

[16] J. Derrick and E. A. Boiten, "Relational Concurrent Refinement," *Formal Asp. Comput.*, vol. 15, no. 2-3, pp. 182–214, 2003. [Online]. Available: http://dblp.uni-trier.de/db/journals/fac/fac15.html#DerrickB03

[17] J. Derrick and E. Boiten, "Relational concurrent refinement part III: Traces, partial relations and automata," *Formal Aspects of Computing*, p. 26, September 2012, accepted for publication. [Online]. Available: http://www.cs.kent.ac.uk/pubs/2012/3251

[18] T. Kahsai and M. Roggenbach, "Property preserving refinement for CSP-CASL," in *WADT*, ser. Lecture Notes in Computer Science, A. Corradini and U. Montanari, Eds., vol. 5486. Springer, 2008, pp. 206–220.

[19] C. Fischer, "CSP-OZ: A combination of Object-Z and CSP," in *FMOODS '97: Proceeding of the IFIP TC6 WG6.1 international workshop on Formal methods for open object-based distributed systems*. London, UK, UK: Chapman & Hall, Ltd., 1997, pp. 423–438.

[20] C. Bolton and J. Davies, "A singleton failures semantics for Communicating Sequential Processes," *Formal Aspects of Computing*, vol. V18, no. 2, pp. 181–210, June 2006. [Online]. Available: http://dx.doi.org/10.1007/s00165-005-0081-x

[21] S. Schneider, "Non-blocking data refinement and traces-divergences semantics," University of Surrey, Technical Report, 2005.

Authorized licensed use limited to: UNIVERSITY MALAYSIA SARAWAK. Downloaded on September 15,2022 at 01:35:14 UTC from IEEE Xplore. Restrictions apply.