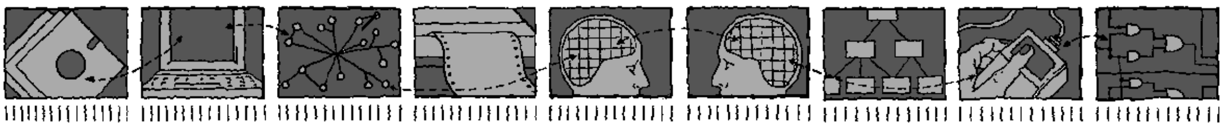
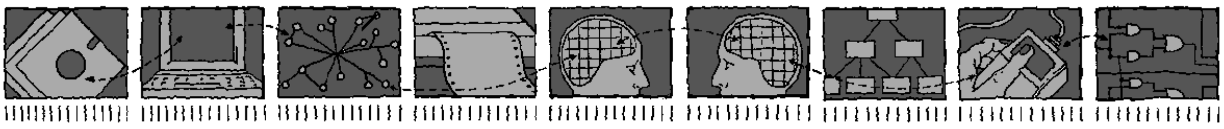


Department of Computing Science and Mathematics
University of Stirling



Construction of LOTOS Behaviour Expressions from Object Communication Diagrams

Robert G. Clark

Technical Report CSM-124

August 1994

Department of Computing Science and Mathematics
University of Stirling

Construction of LOTOS Behaviour Expressions from Object Communication Diagrams

Robert G. Clark

Department of Computing Science and Mathematics, University of Stirling
Stirling FK9 4LA, Scotland

Telephone +44-786-467421, Facsimile +44-786-464551
Email rgc@compsci.stirling.ac.uk

Technical Report CSM-124

August 1994

Abstract

The Rigorous Object-Oriented Analysis (ROOA) method is a systematic development process which takes a set of informal requirements and produces a formal object-oriented analysis model expressed in the standard formal description language LOTOS. An intermediate step in the production of the LOTOS model is the creation of an Object Communication Diagram (OCD) which is a graphical representation of the eventual LOTOS behaviour expression.

It is possible to construct object communication diagrams for which there is no corresponding LOTOS behaviour expression. We propose a condition that must be satisfied by an object communication diagram for there to be a guarantee that there is a corresponding LOTOS behaviour expression and prove that object communication diagrams of arbitrary complexity which satisfy this condition do indeed have a corresponding LOTOS behaviour expression. We then give an algorithm for the construction of a LOTOS behaviour expression from an object communication diagram which satisfies the condition.

1 Introduction

The Rigorous Object-Oriented Analysis (ROOA) method [3, 4] is a systematic development process which takes a set of informal requirements and produces a formal object-oriented analysis model expressed in the standard formal description language LOTOS [1, 2]. The aim of the LOTOS model is to give a complete and accurate description of the problem in terms of entities from the problem domain. It describes both the static and dynamic aspects of a problem and acts as a requirements specification.

An intermediate step in the production of the LOTOS model is the creation of an Object Communication Diagram (OCD). The OCD is a graph in which each node represents an object and each arc connecting two objects represents a gate of communication between them. It is used within ROOA to give a graphical representation of the eventual LOTOS behaviour expression. An important question is the condition(s) that have to be fulfilled to be able to guarantee that an OCD can be represented by a LOTOS behaviour expression.

The gates may only be used in one of two ways:

- Two objects are connected via a gate which is used for no other purpose. We refer to such a gate as a *simple gate* (an *S gate*).
- An object (the *MP object* of the gate) is individually connected with several user objects (*U objects*) via a gate which is used for no other purpose. We refer to such a gate as a *multiple provider* gate (an *MP gate*). A *U object* may itself be the *MP object* for another *MP gate*. A *non MP object* is an object which is not an *MP* object for any *MP* gate.

The message connections between objects in an object model can always be represented by an object communication diagram as defined above. We wish to represent such a diagram by a LOTOS behaviour expression in which each object is represented as a behaviour expression which is the instantiation of a LOTOS process and the objects (LOTOS process instantiations) are connected, through gates, by the LOTOS parallel composition operators.

We define *MP expressions*, *U expressions* and *non MP expressions* to be the LOTOS behaviour expressions which represent *MP* objects, *U* objects and non *MP* objects respectively.

When objects (process instantiations) are created dynamically, the presence of a factory object (process generator) is required, within which the creation takes place. This is described in [4]. The set of generated objects (process instantiations) are then equivalent to a single object (the process generator). A factory object can be treated in exactly the same way as any other object.

As an object communication diagram only supports two-way synchronization, each LOTOS gate may appear in one, and only one, parallel operator in the final behaviour expression. If it is an S gate s connecting the two non MP objects A and B, then it will combine two behaviour expressions, one of which contains the non MP expression for A while the other contains the non MP expression for B. The combined behaviour expression is:

$$\textbf{hide } s \textbf{ in}$$
$$(...A[s,...]...) \mid [s,...] \mid (...B[s,...]...)$$

If it is an MP gate, connecting an MP object A with a set of U objects $U_1, U_2, ...$, then it will combine two behaviour expressions, one of which contains the MP expression for A while the other contains a composition of the U expressions for all the U objects. We are therefore concerned with the bracketing of behaviour expressions into more complex behaviour expressions.

It is possible to construct object communication diagrams for which there is no corresponding LOTOS behaviour expression. Examples are given in Figures 1 and 2.

We propose a condition that must be satisfied by an object communication diagram for there to be a guarantee that there is a corresponding LOTOS behaviour expression and prove that object communication diagrams of arbitrary complexity which satisfy this condition do indeed have a corresponding LOTOS behaviour expression. We then give an algorithm for the construction of a LOTOS behaviour expression from an object communication diagram which satisfies the condition.

2 The Condition

Associated with each *MP* gate g_i is its *MP* object A_i.

We define a *sequence* of *MP* gates to be a set of *MP* gates $g_1, g_2, ..., g_k$ with associated *MP* objects $A_1, A_2, ..., A_k$, where $k > 1$ and in which object A_i is a *U* object for *MP* gate g_{i+1} where $i = 1, 2, ..., k-1$.

We define a *cycle* of *MP* gates to be a sequence of *MP* gates which has the additional property that object A_k is a *U* object for *MP* gate g_1.

We define an *open cycle* of *MP* gates to be a cycle of *MP* gates in which at least one gate g_i has as its *U* objects:

- one, and only one, *MP* object, together with
- a set of one or more non *MP* objects none of which is also a *U* object for another *MP* gate.

A *closed cycle* of *MP* gates is a *cycle* of *MP* gates which is not an *open cycle.*

For it always to be possible to construct a LOTOS behaviour expression corresponding to an object communication diagram, the diagram must not contain a closed cycle of *MP* gates.

3 Example Networks

The object communication diagrams in Figures 1 and 2 do not fulfil the above condition and do not have a corresponding LOTOS behaviour expression. The reasons are given below:

Network in Figure 1:

A_1 is the *MP* object for *MP* gate g_1 and is a *U* object for *MP* gate g_2.
A_2 is the *MP* object for *MP* gate g_2 and is a *U* object for *MP* gate g_1.
We therefore have a cycle of *MP* gates.
B is a *U* object for *MP* gate g_1, but is also a *U* object for *MP* gate g_2.
We therefore have a closed cycle of *MP* gates.

Network in Figure 2:

A_1 is the *MP* object for *MP* gate g_1 and is a *U* object for *MP* gate g_2.
A_2 is the *MP* object for *MP* gate g_2 and is a *U* object for *MP* gate g_3.

A_3 is the *MP* object for *MP* gate g_3 and is a *U* object for *MP* gate g_1.
We therefore have a cycle of *MP* gates.
B is a *U* object for *MP* gates g_1, but is also a *U* object for *MP* gates g_2 and g_3.
We therefore have a closed cycle of *MP* gates.

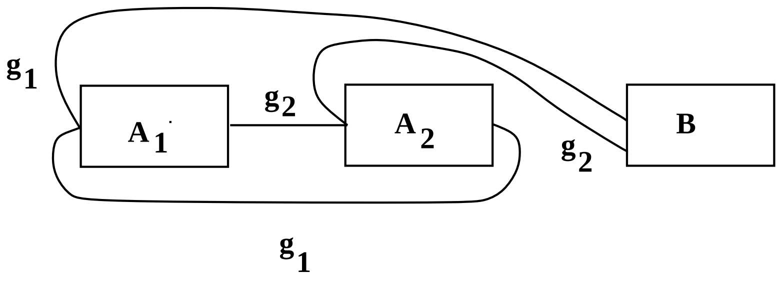


Figure 1: Network with a closed cycle of two *MP* objects A_1 and A_2.

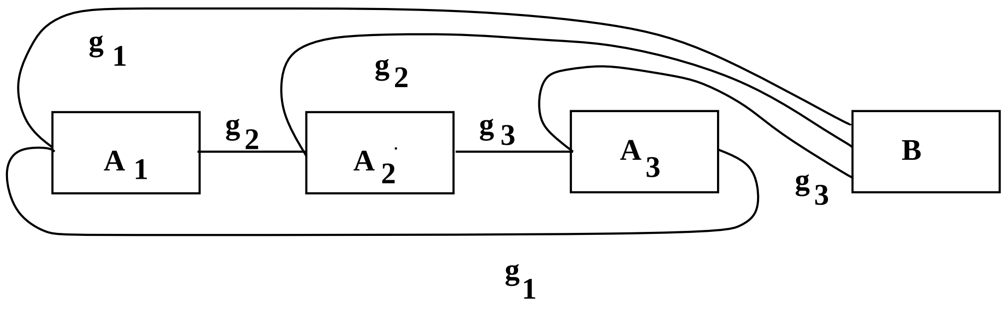


Figure 2: Network with a closed cycle of three *MP* objects.

The object communication diagram in Figure 3 is an open cycle and can be represented by the behaviour expression:

hide g_1, g_2 **in**
$(A_1[g_1, g_2] \,|||\, C[g_2])$
$|[g_1, g_2]|$
(**hide** g_3 **in** $(A_2[g_2, g_3] \,|||\, B[g_1, g_3])\; |[g_3]|\; A_3[g_1, g_3])$

4 The Proof

The Proposition

All object communication diagrams which fulfil the condition have a corresponding LOTOS behaviour expression.

The Proof

Each node in an object communication diagram represents a single object and so it has a corresponding LOTOS behaviour expression which consists of the instantiation of a single LOTOS process. We wish to compose these behaviour expressions into a single behaviour expression which represents the complete system. Composing behaviour expressions is equivalent to combining objects, which have corresponding LOTOS behaviour expressions, into composite objects which also have corresponding LOTOS behaviour expressions.

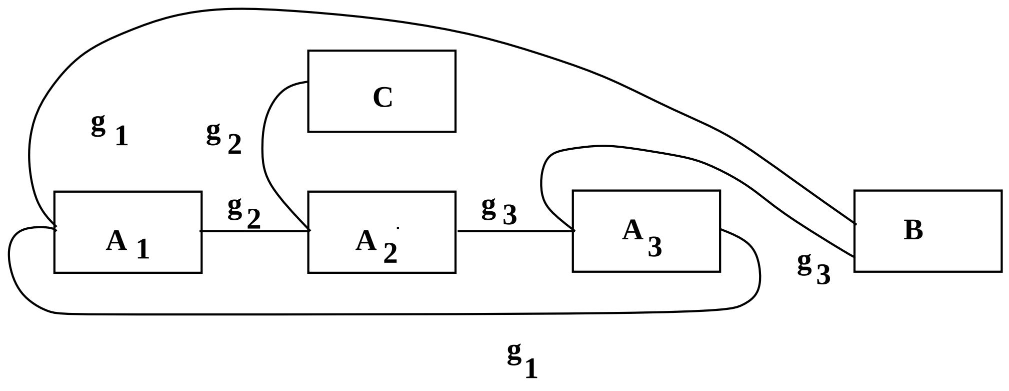


Figure 3: Network with an open cycle of three *MP* objects.

We extend our definition of *MP, U* and non *MP* expressions so that they can also represent composite objects. In the following, when we refer to *objects*, we include the possibility that the objects may be composite.

The proof proceeds by demonstrating that there exists an order in which the objects in an arbitrary object communication diagram, which does not contain a closed cycle of *MP* gates, can be *reduced* to a single composite object. The corollary of this is that the behaviour expression corresponding to the objects can then be composed into a single LOTOS behaviour expression.

We will represent a LOTOS behaviour expression A, which offers synchronization on gates $a_1, ..., a_n$, as $A[a_1, ..., a_n]$.

4.1 Reducing the number of non *MP* objects

Consider any two non *MP* expressions A and B which correspond to non *MP* objects in an object communication diagram. They have the general form:

$$A[a_1, ..., a_m, c_1, ..., c_n, d_1, ..., d_p]$$

and

$$B[b_1, ..., b_q, c_1, ..., c_n, d_1, ..., d_p]$$

where:

- $m, n, p, q = 0, 1, 2, ...,$
- we use the convention that the sequence $x_1, ..., x_k$ represents the empty sequence when $k = 0$,
- both expressions offer synchronization on gates $c_1, ..., c_n, d_1, ..., d_p$, but they only interact with each other on the S gates $c_1, ..., c_n$,
- there is the null intersection between the set of gates $a_1, ..., a_m$ and the set of gates $b_1, ..., b_q$.

As the gates $c_1, ..., c_n$ are S gates, they can only be used in connecting A and B. Hence, when $n > 0$, the two non *MP* expressions A and B can be composed to form the single non *MP* expression:

$$\begin{array}{l} \textbf{hide } c_1, ..., c_n \textbf{ in} \\ \quad A[a_1, ..., a_m, c_1, ..., c_n, d_1, ..., d_p] \mid [c_1, ..., c_n] \mid B[b_1, ..., b_q, c_1, ..., c_n, d_1, ..., d_p] \end{array}$$

and, when $n = 0$, as the single non *MP* expression:

$$A[a_1, ..., a_m, d_1, ..., d_p] \mid\mid\mid B[b_1, ..., b_q, d_1, ..., d_p]$$

which, in both cases, can be represented as the non *MP* expression:

$$AB[a_1, ..., a_m, d_1, ..., d_p, b_1, ..., b_q]$$

This non *MP* expression corresponds to a single (composite) non *MP* object.

Conclusion 1: A pair of non *MP* expressions can always be replaced by an equivalent single non *MP* expression. Hence, all the non *MP* expressions corresponding to the objects in an object communication diagram which only contains non *MP* objects can be composed into a single LOTOS behaviour expression which corresponds to a single non *MP* object.

4.2 Removing a single *MP* gate

When we have an *MP* gate g, which connects its *MP* object A with a set of U objects, all of which are non *MP* objects, then application of **Conclusion 1** reduces this to a gate connecting A with a single non *MP* object as a U object. Gate g is then no longer an *MP* gate.

Conclusion 2: A single *MP* gate, all of whose U objects are non *MP* objects, can be transformed into an S gate by composing the U objects into a single non *MP* object.

4.3 Removing a sequence of *MP* gates

If we have a sequence of k *MP* gates $g_1, g_2, ..., g_k$ with associated *MP* objects $A_1, A_2, ..., A_k$, where $k > 1$ which :

- do not form a cycle and
- are not part of a longer sequence,

then the U objects for gate g_1 must all be non *MP* objects. (If one of the U objects for gate g_1 were also an *MP* object then its *MP* gate would be the first element of a sequence of *MP* gates of length $k + 1$.) Application of **Conclusion 2** means that *MP* gate g_1 can be transformed into an S gate. Hence, a sequence of k *MP* gates which meets the above conditions can always be reduced to a sequence of $k - 1$ *MP* gates.

Conclusion 3: An object communication diagram which contains n *MP* gates, all of which either exist singly or in a sequence which does not form a cycle, may be transformed into an equivalent object communication diagram which contains $n - 1$ *MP* gates all of which either exist singly or in a sequence which does not form a cycle. An object communication diagram, all of whose *MP* gates either exist singly or in a sequence which does not form a cycle, may therefore be transformed into an object communication diagram which contains no *MP* gates and hence no *MP* objects.

From **Conclusion 1**, such a diagram can, in turn, be transformed into an object communication diagram which contains only a single non *MP* object.

4.4 Breaking an Open Cycle

Consider an *open cycle* of *MP* gates $g_1, g_2, ..., g_k$ with associated *MP* objects $A_1, A_2, ..., A_k$, where $k > 1$ and the *MP* gate g_i which has as its *U* objects:

- one, and only one, *MP* object, together with
- a set of one or more non *MP* objects none of which is a *U* object for another *MP* gate.

From **Conclusion 1**, a set of one or more non *MP* objects can be composed into a single non *MP* object. We therefore only have to consider the following structure:

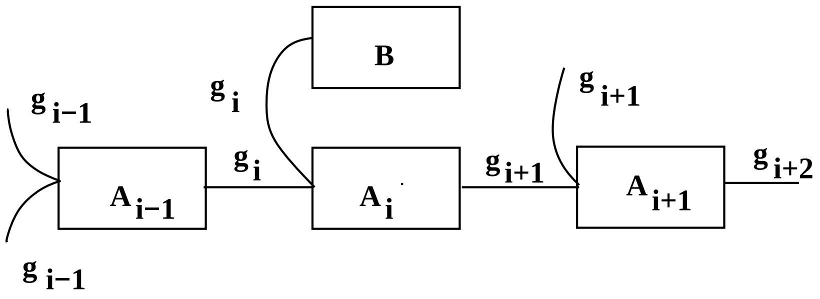


Figure 4: Breaking an Open Cycle.

Objects B and A_{i-1} individually communicate with A_i on gate g_i. Objects B and A_{i-1} may communicate with each other on an S gate, but as that makes no difference to the argument, we shall ignore that possibility. If objects B and A_{i-1}, the U objects of *MP* gate g_i, were combined then gate g_i would no longer be an *MP* gate. By definition, object B cannot communicate with A_{i-1} on an *MP* gate and so they can be combined.

As gate g_i is no longer an MP gate, the sequence $g_1, g_2, ..., g_k$ no longer forms a cycle of *MP* gates. As B is not a U object for any other *MP* gate, another cycle cannot have been created by the amalgamation.

The other problem that must be considered is that the incorporation of the non *MP* object B into an *MP* object might result in another open cycle being transformed into a closed cycle. However as the condition states that object B cannot be a U object for a second *MP* gate, this problem does not arise.

All open cycles can therefore be removed from the diagram and so an object communication diagram which only contains open cycles can always be replaced by an equivalent object communication diagram which contains no cycles.

From **Conclusion 3**, such a diagram can, in turn, be re-written as an object communication diagram which contains only a single non *MP* object.
QED

5 Constructing a LOTOS Behaviour Expression

Constructing a LOTOS behaviour expression corresponding to a large object communication diagram can be a daunting task. The order in which the combinations take place is important as combining non *MP* objects can result in the conversion of an open cycle of *MP* gates into a closed cycle of *MP* gates. For example, if the non *MP* objects B and C in Figure 3 were combined, we would get the position shown in Figure 2. The first simplification should therefore be concerned with the removal of any open cycles from the system.

This paper has given conditions which, if they are satisfied, guarantee that it is possible to construct a behaviour expression from an object communication diagram of arbitrary complexity. It is also the case that some object communication diagrams which do not meet the condition will have a LOTOS behaviour expression, but the addition of further more complex conditions does not seem worthwhile from a practical perspective.

As the proofs proceed by systematically reducing a complex object communication diagram into a simpler object communication diagram and eventually reducing the diagram to a single object, they provide the basis for an algorithm for the construction of a LOTOS behaviour expression which describes the original diagram. The algorithm is given below.

A unique behaviour expression is not generated, but the set of behaviour expressions which can be generated all exhibit exactly the same behaviour; i.e. they are bisimilar.

5.1 The Algorithm

```
if there are any closed cycles of $MP$ gates then
    terminate without having produced a LOTOS behaviour expression
end if
while there are open cycles of $MP$ gates loop
    assertion: There are $n$ open cycles, where $n \geq 1$
    Identify a gate $g$ in the cycle which has as its $U$ objects:
        • one, and only one, $MP$ object,
        • a set of non $MP$ objects, none of which is a
            $U$ object for another $MP$ gate.
    Compose the behaviour expressions corresponding to the $U$ objects
        of gate $g$ into a single $MP$ expression.
    assertion: Gate $g$ is no longer an $MP$ gate and so the open cycle is broken.
    assertion: This cannot create a new open cycle or create a closed cycle.
    assertion: There are $n - 1$ open cycles, where $n \geq 1$
end loop
while there are $MP$ gates loop
    assertion: There are $n$ $MP$ gates, where $n \geq 1$
    assertion: There will be at least one $MP$ gate $g$,
        all of whose $U$ objects are non $MP$ objects.
    Compose the behaviour expressions corresponding to the $U$ objects
        of gate $g$ into a single non $MP$ expression.
    assertion: Gate $g$ is now an $S$ gate.
    assertion: There are $n - 1$ $MP$ gates, where $n \geq 1$
end loop
while there are $S$ gates loop
    assertion: There are $n$ $S$ gates, where $n \geq 1$
    Compose the behaviour expressions corresponding to two non $MP$
        objects connected by an $S$ gate.
    assertion: there are $n - 1$ $S$ gates, where $n \geq 1$
end loop
assertion: We have a single LOTOS behaviour expression.
```

6 An Example

As an example of the construction of a LOTOS behaviour expression, consider the object communication diagram in Figure 5.

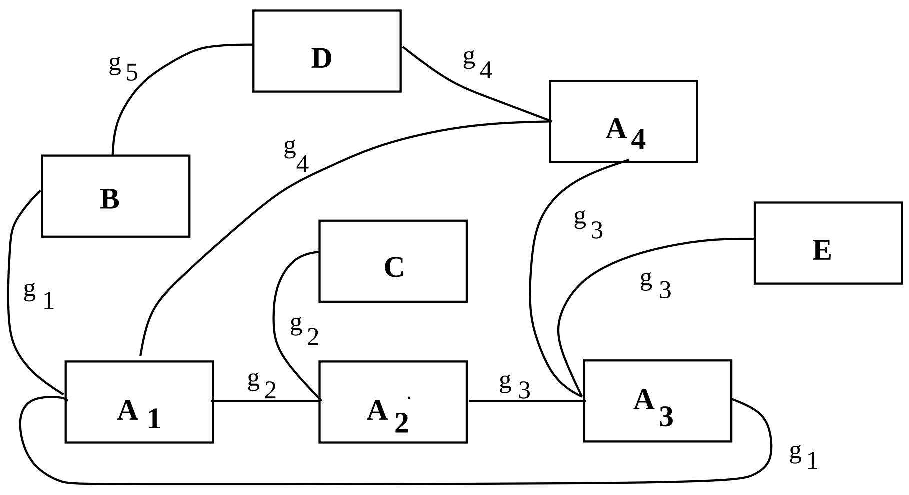


Figure 5: Example Network.

There are no closed cycles.

There are two open cycles, $g_1g_2g_3g_1$ and $g_1g_4g_3g_1$. Let us decide to break cycle $g_1g_2g_3g_1$ first. Consider gate g_2. Its *U* objects are C and A_1. As C is not a *U* object for any other *MP* gate, we can compose C and A_1 giving the *MP* expression:

$$A_1[g_1, g_2, g_4] \mid\mid\mid C[g_2]$$

which we shall refer to as A_1C. Object A_2 is now a non *MP* object, gate g_2 is now an *S* gate and cycle $g_1g_2g_3g_1$ has been broken.

We must now break cycle $g_1g_4g_3g_1$. Consider gate g_4. Its *U* objects are D and A_1C. As D is not a *U* object for any other *MP* gate, we can compose D and A_1C giving the *MP* expression:

$$(A_1[g_1, g_2, g_4] \mid\mid\mid C[g_2]) \mid\mid\mid D[g_4, g_5]$$

which we shall refer to as $(A_1C)D$. Object A_4 is now a non *MP* object, gate g_4 is now an *S* gate and cycle $g_1g_4g_3g_1$ has been broken.

There are now no cycles.

Gates g_1 and g_3 are still *MP* gates. Consider gate g_3. All its *U* objects, A_2, A_4 and E, are non *MP* objects and so their corresponding expressions can be composed to give the non *MP* expression:

$$(A_2[g_2, g_3] \mid\mid\mid E[g_3]) \mid\mid\mid A_4[g_3, g_4]$$

which we shall refer to as $(A_2E)A_4$. Object A_3 is now a non *MP* object and gate g_3 is now an *S* gate.

Now consider gate g_1. Its *U* objects, A_3 and B are non *MP* objects and so their corresponding expressions can be composed to give the non *MP* expression:

$$A_3[g_1, g_3] \mid\mid\mid B[g_1, g_5]$$

which we shall refer to as A_3B. Object $(A_1C)D$ is now a non *MP* object and gate g_1 is now an *S* gate.

There are now no *MP* objects. The non *MP* objects, $(A_1C)D$, $(A_2E)A_4$ and A_3B, can now be combined in any order.

```
hide $g_1, g_3, g_5$ in
    ( hide $g_2, g_4$ in
        $((A_1[g_1, g_2, g_4] \mid\mid\mid C[g_2]) \mid\mid\mid D[g_4, g_5])$
        $\mid [g_2, g_4] \mid$
        $((A_2[g_2, g_3] \mid\mid\mid E[g_3]) \mid\mid\mid A_4[g_3, g_4])$
    )
    $\mid [g_1, g_3, g_5] \mid$
    $(A_3[g_1, g_3] \mid\mid\mid B[g_1, g_5])$
```

7 Conclusion

Conditions for object communication diagrams have been given which, when satisfied, mean that the diagram can be represented by a LOTOS behaviour expression. The proof provides a basis for an algorithm for the construction of LOTOS behaviour expressions.

In a realistic object model, cycles of *MP* gates should seldom arise. Objects usually have services provided by lower level *MP* objects and these lower level servers tend not to be clients of servers higher in the chain. In any case, a sequence of *MP* gates can always be broken by introducing an extra non *MP* object into the sequence to decouple an adjacent pair of *MP* objects.

References

[1] T. Bolognesi and E. Brinksma. Introduction to the ISO Specification Language LOTOS. *Computer Networks and ISDN Systems*, 14(1):25–59, 1987.

[2] E. Brinksma (ed). *Information Processing Systems — Open Systems Interconnection — LOTOS — A Formal Description Technique Based on the Temporal Ordering of Observation Behaviour, ISO 8807*, 1988.

[3] A.M.D. Moreira and R.G. Clark. Rigorous Object-Oriented Analysis. Technical Report CSM-109, Computing Science Department, University of Stirling, Scotland, 1993.

[4] A.M.D. Moreira and R.G. Clark. Combining Object-Oriented Analysis and Formal Description Techniques. In M. Tokoro and R. Pareschi, editors, *8th European Conference on Object-Oriented Programming: ECOOP '94*, LNCS 821, pages 344–364. Springer-Verlag, July 1994.