University of Twente
university for technical and
social sciences

MEMORANDA INFORMATICA-89-18

A VERIFICATION EXERCISE RELATING TO SPECIFICATION STYLES IN LOTOS

Marten van Sinderen

March 1989

Memoranda Informatica
ISSN: 0923-1714
University of Twente,
Department of Computer Science,
P.O.Box 217,
7500 AE Enschede,
The Netherlands.

MEMORANDUM INF-89-18

A VERIFICATION EXERCISE RELATING TO SPECIFICATION STYLES IN LOTOS

Marten van Sinderen

March 1989

University of Twente
Department of Informatics
P.O. Box 217
7500 AE Enschede
The Netherlands

A Verification Exercise Relating to Specification Styles in LOTOS

Marten van Sinderen

University of Twente
Department of Informatics
P.O. Box 217
7500 AE Enschede, NL

Abstract

The design of complex distributed systems can be supported by a methodology where several, increasingly implementation-oriented, formal specifications are successively developed. The relative verification of such specifications is discussed in this memorandum in the context of some example specifications in LOTOS. It is claimed that the adoption of common specification styles can simplify the verification task. Two useful congruence laws which are not contained in the LOTOS standard are presented.

1. Introduction

The notion that the use of formal description techniques (FDTs) for the specification of distributed systems is to be preferred over the use of merely informal methods is gaining more and more support. The successful formal specification of some OSI service and protocol standards (see, for example, [Este89, Loto89]) may have contributed to this trend.

On the other hand, experience with the development of such complex formal specifications (FDs) as those of the OSI standards has shown that the specification job is easily underestimated. One of the major problems turned out to be finding the appropiate structure for a specification, i.e. a structure that expresses functionality at the required level of abstraction, that is comprehensible, and satisfies further stated design objectives. Clearly, agreement on a proper structure for an FD is of prime concern for preserving homogeneity and consistency of the FD if it is produced by a team, a situation we might expect as a rule in an industrial environment.

One way of tackling this problem is by adopting a limited number of *specification styles*, based on specification experience and technical and/or formal arguments. In [Viss88] a design methodology is presented that takes the use of specification styles as its fundament. Particular styles are used to structure specifications that relate to different phases of the system design traject and thus pursue different design objectives. The successive application of the styles is such that in each phase the FD exhibits a refined structure compared with the FD developed in the previous phase. In this way a step-by-step development of a final specification is permitted that will be used as the reference basis for implementation.

An obvious requirement of this approach is that the specifications should be correct w.r.t. each other, i.e. they should describe the same behaviour as the initial (architectural) specification, which, in turn, must correctly capture the users' requirements. This memorandum discusses the relative verification of FDs in the context of some example specifications according to different styles in the FDT LOTOS [ISO8807].

Section 2 presents the example specifications and summarizes the characteristics of the specification styles. The examples are taken from [Viss88], which also includes a more detailed

presentation of each of the styles. In section 3 the relative verification of the examples is presented together with two verification approaches that take advantage of the style-induced specification structures. A brief discussion of the verification exercise is given in section 4. Two annexes are attached: one containing the proofs of two (weak bisimulation) congruence laws which are used in section 3, and the other containing a preliminary investigation of to which extent current service and protocol FD structures support their relative verification, relating to section 4.

2. Examples

The examples presented in this section will be used to illustrate how a few related specification styles can reduce the verification effort in the design of distributed systems. We will only consider the verification of dynamic behaviour, not of data types, which means that the examples can be expressed in basic LOTOS [Bolo87], a subset of standard LOTOS [ISO8807]. Considerations w.r.t. styles for the specification of abstract data types can be found in [Gotz87], while approaches to the verification of data types are contained in, for example, [Gogu78, Ehri85, Orej87].

2.1 Informal specification

The system architecture that is taken as the starting point supports a 'one-time' interaction of two users, Q and A:

- user Q can generate a question which is then transferred by the system to user A;
- in response to the question, user A must generate an answer which is transferred back by the system to user Q.

Figure 1 shows the time-sequence diagram that corresponds to this simple 'question/answer' service.

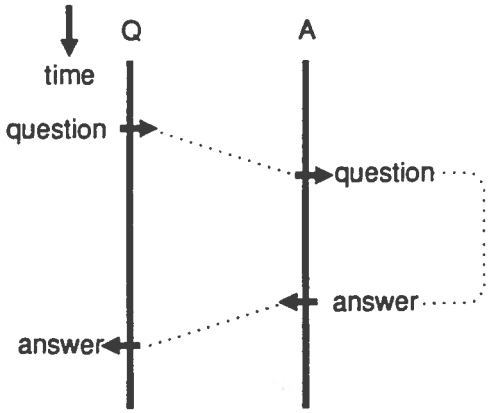


Figure 1: Time-sequence diagram of question/answer service.

2.2 Monolithic specification

In the monolithic style only observable interactions are presented and their temporal ordering relationship is defined as a collection of alternative sequences of interactions.

A monolithic specification is *extensional*, i.e. it does not define internal boundaries of the system. Moreover, a monolithic specification even lacks any useful structure associated with the external boundary which could be used as a basis for possible decomposition. The specification below is therefore a direct, implementation-independent, representation of the question/answer service (synchronization at the gates *Qq*, *Aq*, *Aa*, and *Qa* represent, respectively: acceptance of question, delivery of question, acceptance of answer, and delivery of answer).

```
process Sm[Qq,Aq,Aa,Qa]:noexit := Qq;Aq;Aa;Qa;stop endproc
```

2.3 Constraint-oriented specification

In the constraint-oriented style only observable interactions are presented, but their temporal ordering relationship is composed as a conjunction of separate constraints. Each constraint is defined on the subset of the interaction set that is relevant to it.

Also a constraint-oriented specification is extensional. The external behaviour, however, is structured in terms of constraint specifications that may provide useful hints for internal structuring. The following specification shows a separation of local constraints (process *L*) and remote, or end-to-end, constraints (process *R*). Local constraints apply at each of the user interfaces; remote constraints relate the two user interfaces and can be further decomposed into two one-directional constraint parts (processes *Rq* and *Ra*).

```
process Sc[Qq,Aq,Aa,Qa]:noexit := ( L[Qq,Qa] ||| L[Aq,Aa] ) || R[Qq,Aq,Qa,Aa] where
    process L[Xq,Xa]:noexit := Xq;Xa;stop endproc
    process R[Qq,Aq,Qa,Aa]:noexit := Rq[Qq,Aq] ||| Ra[Aa,Qa] where
        process Rq[Qq,Aq]:noexit := Qq;Aq;stop endproc
        process Ra[Aa,Qa]:noexit := Aa;Qa;stop endproc
    endproc
endproc
```

2.4 Resource-oriented specification

In the resource-oriented style both observable and internal interactions are presented. The temporal ordering relationship of the observable interactions is defined by the composition of separate 'resources' that hides internal interactions. Each resource is defined by a temporal ordering of external and internal interactions or of merely internal interactions.

A resource-oriented specification is called *intensional* since it defines internal boundaries resulting from a composition of the overall specification from specification parts that reflect system resources. The specification below illustrates the provision of the question/answer service by two protocol entities (processes *QE* and *AE*) that interwork via an underlying service (process *US*). The protocol entities are specified, according to the constraint-oriented style, in terms of local (as in section 2.3), send and receive constraints. The underlying service is a simple connectionless service for the transfer of data from *UQs* to *UAr*, or from *UAs* to *UQr*. No relationship between the two directions of transfer is assumed by this service.

```
process Sr[Qq,Aq,Aa,Qa]:noexit :=
    hide UQs,UQr,UAs,UAr in
    ( ( QE[Qq,Qa,UQs,UQr] ||| AE[Aq,Aa,UAs,UAr] ) |[UQs,UQr,UAs,UAr]| US[UQs,UQr,UAs,UAr] )
where
    process QE[Qq,Qa,UQs,UQr]:noexit := L[Qq,Qa] |[Qq,Qa]| ( QEs[Qq,UQs] ||| QEr[Qa,UQr] ) where
        process QEs[Qq,UQs]:noexit := Qq;UQs;stop endproc
        process QEr[Qa,UQr]:noexit := UQr;Qa;stop endproc
    endproc
    process AE[Aq,Aa,UAs,UAr]:noexit := L[Aq,Aa] |[Aq,Aa]| ( AEs[Aa,UAs] ||| AEr[Aq,UAr] ) where
        process AEs[Aa,UAs]:noexit := Aa;UAs;stop endproc
        process AEr[Aq,UAr]:noexit := UAr;Aq;stop endproc
    endproc
    process US[UQs,UQr,UAs,UAr]:noexit := UQs;UAr;stop ||| UAs;UQr;stop endproc
endproc
```

3. Verification

The previous examples present the same system architecture if we can show that:

1) the constraint-oriented specification of the question/answer service is equivalent to the monolithic service specification; and
2) the resource-oriented specification of the question/answer service is equivalent to (i.e., the protocol is correct with respect to) the constraint-oriented service specification.

There are different notions of equivalence: disregarding certain internal events, equating behaviour that cannot be distinguished by classes of finite tests [Abra87], etc. The LOTOS standard contains definitions of, and laws for, two well-established notions of observational equivalence, viz. weak bisimulation equivalence (cf. [Park81]) and testing equivalence (cf. [Nico84, Brin87]). Of the two, weak bisimulation is the strongest, i.e. if two behaviours *B1* and *B2* are weak bisimulation equivalent, they are also testing equivalent, but not necessarily vice versa.

We will demonstrate weak bisimulation equivalence for the two cases mentioned and therefore also their testing equivalence. Use is made of the congruence laws contained in Annex B of the LOTOS standard. A congruence relation is generally stronger than the corresponding equivalence relation: *B1* and *B2* are congruent if, and only if, *C[B1]* and *C[B2]* are equivalent for *all* contexts *C[ ]*. If *B1* and *B2* are weak bisimulation congruent we will write this as *B1* = *B2*. The laws which are used for this exercise are listed in table 1. Three additional laws are included, namely (8), (10) and (11), of which the last two prove useful especially when verifying the correctness of a protocol against its service ((10) and (11) are proven in annex A; the proof of (8) follows directly from the operational semantics of the parallel operator).

In the following, each replacement of a *B1* by a *B2* is accompanied by a reference to the law(s) in table 1 justifying that replacement.

3.1 Verification of constraint-oriented specification against monolithic specification

This verification is straightforward and requires only the application of the laws for instantiation and expansion.

```
Sc[Qq,Aq,Aa,Qa]
= (5)
(Qq;Qa;stop ||| Aq;Aa;stop) || (Qq;Aq;stop ||| Aa;Qa;stop)
= (7)
Qq;( (Qa;stop ||| Aq;Aa;stop) || (Aq;stop ||| Aa;Qa;stop) )
= (7)
Qq;Aq;( (Qa;stop ||| Aa;stop) || (stop ||| Aa;Qa;stop) )
= (7)
Qq;Aq;Aa;( (Qa;stop ||| stop) || (stop ||| Qa;stop) )
= (7)
Qq;Aq;Aa;Qa;stop
= (5)
Sm[Qq,Aq,Aa,Qa]
```

3.2 Verification of resource-oriented specification against constraint-oriented specification

In this case we should exploit the structure of the specifications in order to reformulate the equivalence of two specifications as the congruence of smaller (simpler) behaviours (depending on the context, showing equivalence, instead of congruence, of the partial behaviours may suffice; however, we will not bother with such cases). First, common parts in the specifications may be identified, and subsequently be eliminated if their composition with the remaining part is the same in both specifications. In other words, the equivalence of *C[B,D]* and *C[B,E]*, where *C[ ]* is a context and *B*, *D* and *E* behaviour expressions, is implied by the congruence of *D* and *E*.

Another approach is that of identifying 'corresponding' parts, i.e. behaviours that describe corresponding functionality, and for that reason are expected to be equivalent. If one can identify such parts it is natural to investigate the congruence of the pairs independently: the equivalence of *C[D1,...,Dn]* and *C[E1,...,En]* is implied by the congruence of each of the pairs *D1* and *E1*, *D2* and *E2*, etc.

The structures chosen for the specifications, and therefore also the specification styles adopted, determine how easy or difficult it is to find common/corresponding parts. As stated before, a constraint-oriented specification may provide hints for further (internal) structuring. Following these hints in the development of a resource-oriented specification may thus be advantageous. This is also illustrated in the following.

```
Sr[Qq,Aq,Aa,Qa]
= (5)
hide UQs,UQr,UAs,UAr in
(    (    ( L[Qq,Qa] |[Qq,Qa]| (QEs[Qq,UQs] ||| QEr[Qa,UQr]) )
     |||  ( L[Aq,Aa] |[Aq,Aa]| (AEs[Aa,UAs] ||| AEr[Aq,UAr]) )
     )
|[UQs,UQr,UAs,UAr]|
     US[UQs,UQr,UAs,UAr]
)
= (11)
hide UQs,UQr,UAs,UAr in
(    (    ( L[Qq,Qa] ||| L[Aq,Aa] )
     |[Qq,Qa,Aq,Aa]|
          ( (QEs[Qq,UQs] ||| QEr[Qa,UQr]) ||| (AEs[Aa,UAs] ||| AEr[Aq,UAr]) )
     )
|[UQs,UQr,UAs,UAr]|
     US[UQs,UQr,UAs,UAr]
)
```

```
= (10)
hide UQs,UQr,UAs,UAr in
(     ( L[Qq,Qa] ||| L[Aq,Aa] )
|[Qq,Qa,Aq,Aa]|
      (     ( (QEs[Qq,UQs] ||| QEr[Qa,UQr]) ||| (AEs[Aa,UAs] ||| AEr[Aq,UAr]) )
      |[UQs,UQr,UAs,UAr]|
            US[UQs,UQr,UAs,UAr]
)     )
= (4)
      ( hide UQs,UQr,UAs,UAr in ( L[Qq,Qa] ||| L[Aq,Aa] ) )
|[Qq,Qa,Aq,Aa]|
      (     hide UQs,UQr,UAs,UAr in
            (     ( (QEs[Qq,UQs] ||| QEr[Qa,UQr]) ||| (AEs[Aa,UAs] ||| AEr[Aq,UAr]) )
            |[UQs,UQr,UAs,UAr]|
                  US[UQs,UQr,UAs,UAr]
      )     )
= (1)
      ( L[Qq,Qa] ||| L[Aq,Aa] )
|[Qq,Qa,Aq,Aa]|
      (     hide UQs,UQr,UAs,UAr in
            (     ( (QEs[Qq,UQs] ||| QEr[Qa,UQr]) ||| (AEs[Aa,UAs] ||| AEr[Aq,UAr]) )
            |[UQs,UQr,UAs,UAr]|
                  US[UQs,UQr,UAs,UAr]
      )     )
```

If we compare the latter behaviour expression with that defining the behaviour of *Sc[Qq,Aq,Aa,Qa]*, then we can observe that they have the first part in common (defining the local behaviour at the interfaces) and that the parallel composition applies to the same gates. Hence it suffices to verify the congruence of the remaining part of both expressions (describing the protocol remote behaviour and the service remote behaviour). Let the protocol remote behaviour, i.e. the last hide expression above, be denoted by *Rr[Qq,Aq,Qa,Aa]*.

```
Rr[Qq,Aq,Qa,Aa]
= (5,10,9)
hide UQs,UQr,UAs,UAr in
(     ( (QEs[Qq,UQs] ||| AEr[Aq,UAr]) ||| (QEr[Qa,UQr] ||| AEs[Aa,UAs]) )
|[UQs,UQr,UAs,UAr]|
      (UQs;UAr;stop ||| UAs;UQr;stop)
)
= (11)
hide UQs,UQr,UAs,UAr in
(     ( (QEs[Qq,UQs] ||| AEr[Aq,UAr]) |[UQs,UAr]| UQs;UAr;stop )
|||
      ( (QEr[Qa,UQr] ||| AEs[Aa,UAs]) |[UQr,UAs]| UAs;UQr;stop )
)
= (4,5)
      ( hide UQs,UQr,UAs,UAr in ( (Qq;UQs;stop ||| UAr;Aq;stop) |[UQs,UAr]| UQs;UAr;stop ) )
|||   ( hide UQs,UQr,UAs,UAr in ( (UQr;Qa;stop ||| Aa;UAs;stop) |[UQr,UAs]| UAs;UQr;stop ) )
= (7,3)
      Qq;( hide UQs,UQr,UAs,UAr in ( (UQs;stop ||| UAr;Aq;stop) |[UQs,UAr]| UQs;UAr;stop ) )
|||   Aa;( hide UQs,UQr,UAs,UAr in ( (UQr;Qa;stop ||| UAs;stop) |[UQr,UAs]| UAs;UQr;stop ) )
= (7,2)
      Qq;i;( hide UQs,UQr,UAs,UAr in ( (stop ||| UAr;Aq;stop) |[UQs,UAr]| UAr;stop ) )
|||   Aa;i;( hide UQs,UQr,UAs,UAr in ( (UQr;Qa;stop ||| stop) |[UQr,UAs]| UQr;stop ) )
= (7,2)
      Qq;i;i;( hide UQs,UQr,UAs,UAr in ( (stop ||| Aq;stop) |[UQs,UAr]| stop ) )
|||   Aa;i;i;( hide UQs,UQr,UAs,UAr in ( (Qa;stop ||| stop) |[UQr,UAs]| stop ) )
```

= (7,3,8)
*Qq;i;i;Aq;**stop** ||| Aa;i;i;Qa;**stop***
= (6,5)
Rq[Qq,Aq] ||| Ra[Qa,Aa]
= (5)
R[Qq,Aq,Qa,Aa]

---

hiding

(1) ***hide** S **in** B = B* if $L(B) \cap S = \emptyset$

(2) ***hide** S **in** g;B = i;(**hide** S **in** B)* if $g \in S$

(3) ***hide** S **in** g;B = g;(**hide** S **in** B)* if $g \notin S$

(4) ***hide** S **in** (B1 ||[S']|| B2) = (**hide** S **in** B1) ||[S']|| (**hide** S **in** B2)* if $S \cap S' = \emptyset$

instantiation

(5) *b[a1,...,an] = Bb[a1/g1,...an/gn]* if **process** *b[g1,...gn]:f := Bb **endproc*** is the format of the corresponding process abstraction for the process-identifier *b*

internal action

(6) *a;i;B = a;B*

expansion theorem

Let *B1 [] B2 [] ... [] Bn* be written as *[]{B1,B2,...,Bn}*, with *n* finite, and let *B = []{bi;Bi | i∈ I}* and *C = []{cj;Cj | j∈ J}*, with *I* and *J* finite sets. Then:

(7) *B|[S]|C = []{bi;(Bi|[S]|C) | bi∉ S, i∈ I} [] [] {cj;(B|[S]|Cj) | cj∉ S, j∈ J} [] [] {a;(Bi|[S]|Cj) | a=bi=cj, a∈ S, i∈ I, j∈ J}*

parallel

(8) ***stop** ||[S]|| B = **stop*** if $L(B) \subseteq S$

(9) *A ||[S]|| B = B ||[S]|| A*

(10) *(A ||[S1]|| B) ||[S2]|| C = A ||[S1]|| (B ||[S2]|| C)* if $L(A) \cap S2 \subseteq L(A) \cap S1$ and $L(C) \cap S1 \subseteq L(C) \cap S2$

(11) *(A|[S1]|B) ||| (C|[S2]|D) = (A|||C) ||[S1∪S2]|| (B|||D)* if $(L(A) \cup L(B)) \cap S2 = \emptyset$, $(L(C) \cup L(D)) \cap S1 = \emptyset$

Table 1: Some useful laws for weak bisimulation congruence (without value expressions).

4. Discussion

The sequence of the transformation steps above is suggested by the fact that it may be possible to find expressions, say *Rrq* and *Rra*, corresponding to the independent behaviours that make up the remote constraints of the constraint-oriented specification, viz. *Rq* and *Ra*. This is indeed possible, and the relative verification of the corresponding parts is then simple (note that the independent verifications are performed simultaneously).

From this it follows that, given a resource- and a constraint-oriented specification that are structured as our example specifications, however with arbitrary complex parts (resources and constraints, respectively), the equivalence of the two specifications can be verified by verifying *Rrq = Rq* and *Rra = Ra*.

It appears that law (11) plays a crucial role in the verification: first it is used to separate the local from the remote constraints in the composed behaviour of the two protocol entities and the underlying service, and subsequently it allows to separate the constraints associated with the two directions of transfer in the protocol remote behaviour. We may wonder whether this approach is still viable if the specifications are not as simple as in the previous case. For example, in the constraint-oriented specification the remote constraints for the two directions of transfer may be interrelated (this is, for example, the case in the OSI session service specification [Sind88]) and

similarly the send and receive actions of a protocol entity in the resource-oriented specification (this is normally the case). For such cases a generalized form of law (11) can be used with similar specification parts as those of our examples, however with more complex parallel compositions (annex B contains a preliminary investigation of exploiting general specification structures for verification purposes). The verification then reduces to proving the congruence:

hide *P∪B* ***in*** *(PR1 ||[P]|| PR2) = SR1 ||[T]|| SR2*, where

PR1 and *PR2*: the protocol remote constraints associated with one direction of transfer and those associated with the other (however, without the hiding gates in *P*);
SR1 and *SR2*: the service remote constraints associated with one direction of transfer and those associated with the other;
P: a set of gates internal to the protocol entities;
T: a set of external gates that are needed to synchronize *SR1* and *SR2*;
B: a set of gates used by the protocol entities to synchronize with the underlying service.

Here, one cannot prove the congruence by verifying *PR1 = SR1* and *PR2 = SR2*, since *PR1* and *PR2* synchronize at a (set of) hidden gate(s). Such a proof of equivalence in general depends on the structure of the parts which may give further structure to the proof.

A final remark should be made on the internal structuring of a system (e.g., the question/answer service). Intuitively we know that this can, in principle, be done in an infinite number of ways. Laws (2) and (6), in particular, illustrate how different decompositions of a system may provide equivalent externally observable behaviour. Law (2) states that internal actions can be replaced by internal events in the behaviours of the compositions; law (6) states that certain internal events in the resulting behaviours cannot be distinguished from the same behaviours without the internal events (there are also laws concerned with internal actions in other contexts). Hence, application of these laws make the particular internal structures expressed by different (e.g. constraint-oriented) specifications disappear and allow them to be transformed, with the help of the other laws, into the same (e.g., constraint-oriented) specification.

References

[Abra87] S. Abramsky, "Observation Equivalence as a Testing Equivalence", Th. Comput. Sci. 53 (1987) 225-241.

[Bolo87] T. Bolognesi, E. Brinksma, "Introduction to the ISO Specification Language LOTOS", Computer Networks and ISDN Systems, Vol. 14, No. 1 (1987) 25-59.

[Brin87] E. Brinksma, G. Scollo, C. Steenbergen, "LOTOS Specifications, their Implementations and their Tests", Proc. IFIP WG6.1, Protocol Specification, Testing, and Verification VI, Montreal, Canada, June 1986 (North-Holland 1987) 349-360.

[DeNi84] R. De Nicola, M.C.B. Hennessy, "Testing Equivalences for Processes", Th. Comput. Sci. 34 (1984) 83-133.

[Ehri85] H. Ehrig, B. Mahr, "Fundamentals of Algebraic Specification 1" (Springer-Verlag 1985).

[Este89] M. Diaz et al. (Eds.), "The Formal Description Technique Estelle, Results of the ESPRIT/SEDOS Project" (North-Holland 1989).

[Gogu78] J.A. Goguen, J.W. Thatcher, E.G. Wagner, "An Initial Algebra Approach to the Specification, Correctness, and Implementation of Abstract Data Types", in: R. Yeh (Ed.), Current Trents in Programming Methodology IV (Prentice-Hall 1978) 80-149.

[Gotz87] R. Gotzhein, "Specifying Abstract Data Types with LOTOS", Proc. IFIP WG6.1, Protocol Specification, Testing, and Verification VI, Montreal, Canada, June 1986 (North-Holland 1987) 13-22.

[ISO8807] ISO, "LOTOS - A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour", Int. Standard ISO 8807, 1989.

[Lage88] J. v.d. Lagemaat, G. Scollo, "On the Use of LOTOS for the Formal Description of a Transport Protocol", Proc. FORTE 88, Formal Description Techniques I, Stirling, Scotland, Sept. 1988 (North-Holland, 1989) 247-261.

[Loto89] P.H.J. van Eijk et al. (Eds.), "The Formal Description Technique LOTOS, Results of the ESPRIT/SEDOS Project" (North-Holland 1989).

[Miln80] R. Milner, "A Calculus of Communicating Systems" (Springer-Verlag 1980).

[Orej87] F. Orejas, "A Characterization of Passing Compatibility for Parameterized Specifications", Th. Comput. Sci. 51 (1987) 205-214.

[Park81] D. Park, "Concurrency and Automata on Infinite Sequences", Proc. 5th GI Conference, LNCS 104 (Springer-Verlag 1981).

[Scol87] G. Scollo, M. v. Sinderen, "On the Architectural Design of the Formal Specification of the Session Standards in LOTOS", Proc. IFIP WG6.1, Protocol Specification, Testing, and Verification VI, Montreal, Canada, June 1986 (North-Holland 1987) 3-14.

[Sind88] M. v. Sinderen, I. Ajubi, F. Caneschi, "The Application of LOTOS for the Formal Description of the ISO Session Layer", Proc. FORTE 88, Formal Description Techniques I, Stirling, Scotland, Sept. 1988 (North-Holland, 1989) 263-277.

[Viss88] C.A. Vissers, G. Scollo, and M. v. Sinderen, "Architecture and Specification Style in Formal Descriptions of Distributed Systems", Proc. IFIP WG6.1, Protocol Specification, Testing, and Verification VIII, Atlantic City, USA, June 1988 (North-Holland 1987) 189-204.

Annex A. Proofs

To prove the congruences (10) and (11) of table 1, we recall the operational semantics of the LOTOS parallel operator as defined by the inference rules that express the transition possibilities of a behaviour expression *B1* |[*S*]| *B2*:

B1 -*g*-> *B1'* and *g*∈ L(*B1*) - *S* => *B1* |[*S*]| *B2* -*g*-> *B1'* |[*S*]| *B2*
B2 -*g*-> *B2'* and *g*∈ L(*B2*) - *S* => *B1* |[*S*]| *B2* -*g*-> *B1* |[*S*]| *B2'*
B1 -*g**-> *B1'* and *B2* -*g**-> *B2'* and *g**∈ (L(*B1*)∩L(*B2*)∩*S*)∪{δ} => *B1* |[*S*]| *B2* -*g*-> *B1'* |[*S*]| *B2'*

Further, we use the notion of (strong) bisimulation equivalence [Park81], applied to LOTOS, which is stronger than weak bisimulation congruence:

B1 ~ *B2* iff a relation *R* over behaviour expressions exists with <*B1*,*B2*>∈ *R*, such that ∀<*A*,*B*>∈ *R* and ∀*g*∈ *G*∪{*i*,δ}, where *G* the set of user-definable gates
(i) *A* -*g*-> *A'* => ∃*B'*. *B* -*g*-> *B'* and <*A'*,*B'*>∈ *R*
(ii) *B* -*g*-> *B'* => ∃*A'*. *A* -*g*-> *A'* and <*A'*,*B'*>∈ *R*

We prove now the strong bisimulation equivalences corresponding to (10) and (11), and therefore also the weak bisimulation congruences.

Theorem
(A |[S1]| B) |[S2]| C ~ *A |[S1]| (B |[S2]| C)* if L(*A*)∩*S2* ⊆ L(*A*)∩*S1* and L(*C*)∩*S1* ⊆ L(*C*)∩*S2*

Proof
Let *B1* =df *(A |[S1]| B) |[S2]| C*, *B1* =df *A |[S1]| (B |[S2]| C)*, and
R =df {<*(E |[S1]| F) |[S2]| G*, *E |[S1]| (F |[S2]| G)*> | *E*,*F*,*G* behaviour expressions}.

From the definition of *R* it follows that <*B1*,*B2*>∈ *R*. We enumerate the potential transition cases of *B1* and *B2* in terms of transitions of *A*, *B* and *C*, and state the conditions such that *B1* -*g*-> *B1'* => ∃*B2'*. *B2* -*g*-> *B2'* and *B2* -*g*-> *B2'* => ∃*B1'*. *B1* -*g*-> *B1'*. The cases and conditions follow from the inference rules defining the operational semantics of the parallel operator:

(a) *A* -*g*-> *A'*: L(*A*) - (*S1*∪*S2*) = L(*A*) - *S1* <=> L(*A*)∩*S2*⊆L(*A*)∩*S1* ;
(b) *B* -*g*-> *B'*: no conditions, i.e. transitions of *B1* and *B2* depend in the same way on L(*A*), L(*B*), L(*C*), *S1*, and *S2* ;
(c) *C* -*g*-> *C'*: L(*C*) - *S2* = L(*C*) - (*S1*∪*S2*) <=> L(*C*)∩*S1*⊆L(*C*)∩*S2* ;
(d) *A* -*g*-> *A'* and *B* -*g*-> *B'*: no conditions;
(e) *A* -*g*-> *A'* and *C* -*g*-> *C'*: (L(*A*)∩L(*C*)∩*S2*) - *S1* = (L(*A*)∩L(*B*)∩*S1*) - *S2*. This condition is only true if the left- and righthand side are equal to ∅ (in which case this transition cannot occur), which is implied by L(*A*)∩*S2*⊆L(*A*)∩*S1* and L(*C*)∩*S1*⊆L(*C*)∩*S2* ;
(f) *B* -*g*-> *B'* and *C* -*g*-> *C'*: no conditions;
(g) *A* -*g*-> *A'* and *B* -*g*-> *B'* and *C* -*g*-> *C'*: no conditions.

It follows directly from the definition of the parallel operator that also <*B1'*,*B2'*>∈ *R*.
QED.

Note that the conditions L(*A*)∩*S2*⊆L(*A*)∩*S1* and L(*C*)∩*S1*⊆L(*C*)∩*S2* of the theorem are the least restrictive conditions under which associativity of the parallel operator holds. Special cases of the theorem, that were also used in section 3, are *(A |[S]| B) |[S]| C = A |[S]| (B |[S]| C) = A |[S]| B |[S]| C*, and *(A |[S1]| B) |[S2]| C = A |[S1]| (B |[S2]| C)* if L(*A*)∩*S2*=∅ and L(*C*)∩*S1*=∅.

Theorem

(A|[S1]|B) |[I1∪I2]| (C|[S2]|D) ~ (A|[I1]|C) |[S1∪S2]| (B|[I2]|D) if L(*A*)∩(*S2*∪*I2*)=∅, L(*B*)∩(*S2*∪*I1*)=∅, L(*C*)∩(*S1*∪*I2*)=∅, and L(*D*)∩(*S1*∪*I1*)=∅

Proof

Let *B1* =df *(A|[S1]|B) |[I1∪I2]| (C|[S2]|D)*, *B2* =df *(A|[I1]|C) |[S1∪S2]| (B|[I2]|D)*, and *R* =df {<*(E|[S1]|F) |[I1∪I2]| (G|[S2]|H)*, *(E|[I1]|F) |[S1∪S2]| (G|[I2]|H)*> | *E,F,G,H* behaviour expressions}.

Clearly, <*B1,B2*>∈ *R*. We check whether for all possible transition cases of *B1* and *B2* the conditions that must apply if *B1 -g-> B1'* => ∃*B2'*. *B2 -g-> B2'* and *B2 -g-> B2'* => ∃*B1'*. *B1 -g-> B1'* are implied by the conditions of the theorem:

(a) *A -g-> A'*: L(*A*) - (*S1*∪*I1*∪*I2*) = L(*A*) - (*S1*∪*S2*∪*I1*). This is true since L(*A*)∩(*S2*∪*I2*)=∅ ;
(b) *B -g-> B'*: L(*B*) - (*S1*∪*I1*∪*I2*) = L(*B*) - (*S1*∪*S2*∪*I2*). True, since L(*B*)∩(*S2*∪*I1*)=∅ ;
(c) *C -g-> C'*: L(*C*) - (*S2*∪*I1*∪*I2*) = L(*C*) - (*S1*∪*S2*∪*I1*). True, since L(*C*)∩(*S1*∪*I2*)=∅ ;
(d) *D -g-> D'*: L(*D*) - (*S2*∪*I1*∪*I2*) = L(*D*) - (*S1*∪*S2*∪*I2*). True, since L(*D*)∩(*S1*∪*I1*)=∅ ;
(e) *A -g-> A'* and *B -g-> B'*: (L(*A*)∩L(*B*)∩*S1*) - (*I1*∪*I2*) = (L(*A*)∩L(*B*)∩(*S1*∪*S2*)) - (*I1*∪*I2*). True, since L(*A*)∩*S2*=∅ ;
(f) *A -g-> A'* and *C -g-> C'* : (L(*A*)∩L(*C*)∩(*I1*∪*I2*)) - (*S1*∪*S2*) = (L(*A*)∩L(*C*)∩*I1*) - (*S1*∪*S2*). True, since L(*A*)∩*I2*=∅ ;
(g) *B -g-> B'* and *D -g-> D'* : (L(*B*)∩L(*D*)∩(*I1*∪*I2*)) - (*S1*∪*S2*) = (L(*B*)∩L(*D*)∩*I2*) - (*S1*∪*S2*). True, since L(*B*)∩*I1*=∅ ;
(h) *C -g-> C'* and *D -g-> D'*: (L(*C*)∩L(*D*)∩*S2*) - (*I1*∪*I2*) = (L(*C*)∩L(*D*)∩(*S1*∪*S2*)) - (*I1*∪*I2*). True, since L(*C*)∩*S1*=∅ ;
(i) *A -g-> A'* and *B -g-> B'* and *C -g-> C'* and *D -g-> D'*:
(L(*A*)∩L(*B*)L(*C*)∩L(*D*)∩*S1*∩*S2*∩(*I1*∪*I2*)) ∪ {δ} =
(L(*A*)∩L(*B*)L(*C*)∩L(*D*)∩(*S1*∪*S2*)∩(*I1*∪*I2*) ∪ {δ}. True, since L(*A*)∩(*S2*∪*I2*)=∅ .

Other transition cases are not possible for *B1* and *B2* with the conditions of the theorem: transitions of *B1* (*B2*, respectively) require a *g* action with *g* ∈ *I1*∪*I2* (*g* ∈ *S1*∪*S2*), while the parts involved are such that some have no gates in common with *I1* (*S1*) and the others have no gates in common with *I2* (*S2*).

It follows directly from the definition of the parallel operator that also <*B1',B2'*>∈ *R*.
QED.

Note that in this case we did not bother to find the least restrictive conditions under which the stated congruence holds. Therefore, the theorem may presumably be formulated with more general conditions. A special case of the theorem, used in section 3, is *(A|[S1]|B) ||| (C|[S2]|D) = (A|||C) |[S1∪S2]| (B|||D)* if (L(*A*)∪L(*B*))∩*S2*=∅ and (L(*C*)∪L(*D*))∩*S1*=∅.

Annex B. Specification Structures

In the verification of a protocol against its intended service, it is desirable to be able to immediately replace the required equivalence relation by one or more simpler ones, by making use of the structures (that explicitly show the commonalities) of the service and protocol FDs. This would be possible if commonly agreed specification styles, as opposed to ad-hoc styles, were used to give the descriptions their suitable structure.

In the following, we are concerned with investigating what would be the simpler required equivalence relations if we agree to use the constraint-oriented style for service FDs and the resource- (constraint-) oriented style for protocol FDs. Both of these styles have been used in the recent past for describing relatively complex system parts, and were considered useful in producing well-structured, hence comprehensible, specifications (see, e.g., [Scol87, Lage88, Sind88, Loto89]). We will assume specification structures which are expected, on basis of our specification experience, to be general (i.e., with which different types of services and protocols can be described) and useful (i.e., such that comprehensibility is improved when compared to, for example, monolithic specifications).

We will make use of the weak bisimulation congruence laws contained in Annex B of the LOTOS standard, the two laws of annex A above ((2) and (4) in the table below), and two additional laws ((1) and (3) in the table below, which can easily be proven in the same way as (4)).

---

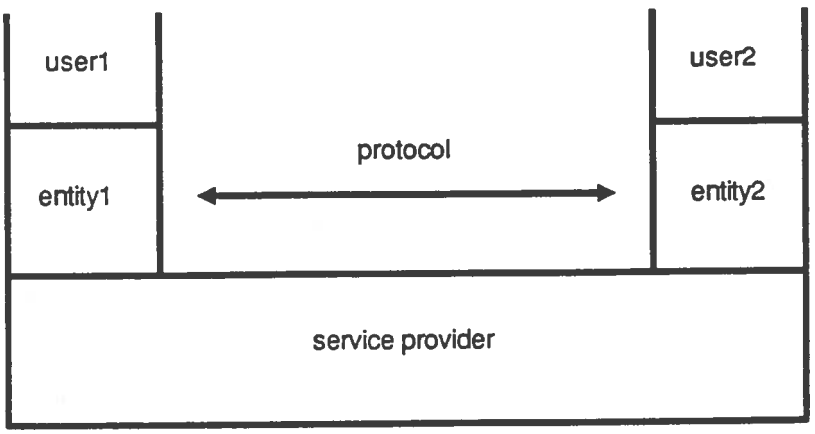
(1) $B \;|[S]|\; B = B$ if $L(B)=S$

(2) $A \;|[S1]|\; (B \;|[S2]|\; C) = (A \;|[S1]|\; B) \;|[S2]|\; C$ if $L(A)\cap S2 \subseteq L(A)\cap S1$ and $L(C)\cap S1 \subseteq L(C)\cap S2$

(3) $A \;|[S1]|\; (B \;|[S2]|\; C) = (A \;|||\; B) \;|[S1\cup S2]|\; C$ if $L(A)\cap S2=\emptyset$ and $L(A)\cap L(B)\cap S1=\emptyset$

(4) $(A \;|[S1]|\; B) \;|[I1\cup I2]|\; (C \;|[S2]|\; D) = (A \;|[I1]|\; C) \;|[S1\cup S2]|\; (B \;|[I2]|\; D)$ if
$L(A)\cap(S2\cup I2)=\emptyset$, $L(B)\cap(S2\cup I1)=\emptyset$, $L(C)\cap(S1\cup I2)=\emptyset$ and $L(D)\cap(S1\cup I1)=\emptyset$

Consider a protocol level in a distributed system as illustrated by the figure below. The protocol consists of two cooperating protocol entities, 'entity1' and 'entity2', which exchange protocol data units (PDUs) via a bottom service (the service provided by the service provider in the figure) in order to provide a top service that supports two coordinating users, 'user1' and 'user2'.



If the bottom service is a communication service, e.g. the OSI transport service, it can generally be described in the constraint-oriented style as:

BS =df *(BL1 ||| BL2) ||[B]|| (BR1 ||| BR2)*
where
B the set of bottom service gates {*bs1,br1,bs2,br1*},
BL1 the local constraints at the entity1 interface,
BL2 the local constraints at the entity2 interface,
BR1 the remote constraints related to data transfer from entity1 to entity2,
BR2 the remote constraints related to data transfer from entity2 to entity1, and
L(*BL1*) =df {*bs1,br1*}, L(*BL2*) =df {*bs2,br1*}, L(*BR1*) =df {*bs1,br2*}, L(*BR2*) =df {*bs2,br1*}.

Notice that the interfaces are each represented by two gates. At one of the two gates - say *bs1* at the entity1 interface and *bs2* at the entity2 interface - request and response (service primitive) events occur, and at the other gate indication and confirm events.

The top service may be more complex than the bottom service and for that reason may be described, again in the constraint-oriented style, as:

TS =df *(TL1 ||| TL2) ||[T]|| (TR1 ||[T]|| TR2)*
where
T the set of top service gates {*ts1,tr1,ts2,tr2*},
TL1 the local constraints at the user1 interface,
TL2 the local constraints at the user2 interface,
TR1 the remote constraints related to data transfer from user1 to user2,
TR2 the remote constraints related to data transfer from user2 to user1, and
L(*TL1*) =df {*ts1,tr1*}, L(*TL2*) =df {*ts2,tr2*}, L(*TR1*) =df L(*TR2*) =df *T*.

As opposed to the bottom service behaviour, the above expression allows the remote constraints for both directions of transfer to be interrelated. The choice of the service gates is similar to that for the bottom service description.

The protocol entities may then be described in the resource-constraint-oriented style as follows:

PE1 =df ***hide*** *P1* ***in*** *((TL1 ||| BL1) ||[T1∪B1]|| (PEs1 ||[P1]|| PEr1))* and
PE2 =df ***hide*** *P2* ***in*** *((TL2 ||| BL2) ||[T2∪B2]|| (PEs2 ||[P2]|| PEr2))*
where
T1 =df {*ts1,tr1*}, *T2* =df {*ts2,tr2*}, *B1* =df {*bs1,br1*}, *B2* =df {*bs2,br2*},
P1 a set of internal gates used in the definition of entity1,
P2 a set of internal gates used in the definition of entity2,
PE1s the constraints on sending PDUs by entity1,
PE1r the constraints on receiving PDUs by entity1,
PE2s the constraints on sending PDUs by entity2,
PE2r the constraints on receiving PDUs by entity2,
TL1, *TL2*, *BL1* and *BL2* as above, and
L(*PEs1*) =df {*ts1,bs1*}∪*P1*, L(*PEr1*) =df {*tr1,br1*}∪*P1*, L(*PEs2*) =df {*ts2,bs2*}∪*P2*, L(*PEr2*) =df {*tr2,br2*}∪*P2*.

To prove the correctness of the protocol it is necessary to verify that:

TS = ***hide*** *B* ***in*** *((PE1 ||| PE2) ||[B]|| BS)*

First we investigate the composition of the protocol entities and the bottom service. Since the expression *BL1 ||| BL2* occurs twice in the composition, one obvious aim is to eliminate one of these occurrences.

Let *PEa1* =df ***hide*** *P1* ***in*** *(PEs1 ||[P1]|| PEr1)* and *PEa2* =df ***hide*** *P2* ***in*** *(PEs2 ||[P2]|| PEr2)*, then:

(PE1 ||| PE2) ||[B]|| BS
= (instantiation, hiding)
(((TL1 ||| BL1) ||[T1∪B1]|| PEa1) ||| ((TL2 ||| BL2) ||[T2∪B2]|| PEa2))
||[B]||
((BL1 ||| BL2) ||[B]|| (BR1 ||| BR2))
= (4)
((PEa1 ||| PEa2) ||[T∪B]|| ((TL1 ||| BL1) ||| (TL2 ||| BL2)))
||[B]||
((BL1 ||| BL2) ||[B]|| (BR1 ||| BR2))
= (3)
((PEa1 ||| PEa2) ||[T]|| (TL1 ||| TL2))
||[B]||
(BL1 ||| BL2)
||[B]||
(BL1 ||| BL2)
||[B]||
(BR1 ||| BR2)
= (1)
((PEa1 ||| PEa2) ||[T]|| (TL1 ||| TL2))
||[B]||
(BL1 ||| BL2)
||[B]||
(BR1 ||| BR2)

Hence, if we have a composition of a protocol and its bottom service with the above structures, we may eliminate the local constraints related to the bottom service in either of the components.

Now we can check whether it is possible to simplify the equivalence relation. We start with trying to transform the above expression such that it consists of a parallel composition of the local constraints related to the top service and the protocol remote constraints.

((PEa1 ||| PEa2) ||[T]|| (TL1 ||| TL2)) ||[B]|| (BL1 ||| BL2)
||[B]||
(BR1 ||| BR2)
= (2)
(((PEa1 ||| PEa2) ||[B]|| (BL1 ||| BL2)) ||[T]|| (TL1 ||| TL2))
||[B]||
(BR1 ||| BR2)
= (2)
((PEa1 ||| PEa2) ||[B]|| (BL1 ||| BL2) ||[B]|| (BR1 ||| BR2))
||[T]||
(TL1 ||| TL2)

Hiding the internal gates in the latter expression yields:

hide** B **in
(((PEa1 ||| PEa2) |[B]| (BL1 ||| BL2) |[B]| (BR1 ||| BR2)) |[T]| (TL1 ||| TL2))
=
(TL1 ||| TL2)
|[T]|
*(**hide** B **in** ((PEa1 ||| PEa2) |[B]| (BL1 ||| BL2) |[B]| (BR1 ||| BR2)))*

Since the same instance of the parallel operator is used in the above composition as in the parallel composition of the local and remote constraints in the top service description, we may replace the required equivalence relation by:

*TR1 |[T]| TR2 = **hide** B **in** ((PEa1 ||| PEa2) |[B]| (BL1 ||| BL2) |[B]| (BR1 ||| BR2))*

thus eliminating the local constraints related to the top service.

Next, we may try to find expressions in the protocol remote constraints that correspond to the 'one-directional' service remote constraints, *TR1* and *TR2*, respectively.

Let *Bs* =df {*bs1*,*br2*} and *Br* =df {*bs2*,*br1*}, and suppose that we can separate the send and receive concerns in the local constraints related to the bottom service, i.e.:

BL1 =df ***hide** I1 **in** (BLs1 |[I1]| BLr1)*, *BL2* =df ***hide** I2 **in** (BLs2 |[I2]| BLr2)*

Further, let *I* =df *I1*∪*I2* and *P* =df *P1*∪*P2*.

***hide** B **in** ((PEa1 ||| PEa2) |[B]| (BL1 ||| BL2) |[B]| (BR1 ||| BR2))*
=
hide** B∪P **in
(
((PEs1 |[P1]| PEr1) ||| (PEs2 |[P2]| PEr2))
|[B]|
*((**hide** I1 **in** (BLs1 |[I1]| BLr1)) ||| (**hide** I2 **in** (BLs2 |[I2]| BLr2)))*
|[B]|
(BR1 ||| BR2)
)
= (4)
hide** B∪P∪I **in
(
((PEs1 ||| PEr2) |[P]| (PEs2 ||| PEr1))
|[B]|
((BLs1 ||| BLr2) |[I]| (BLs2 ||| BLr1))
|[B]|
(BR1 ||| BR2)
)
= (4)
hide** B∪P∪I **in
(
(((PEs1 ||| PEr2) |[Bs]| (BLs1 ||| BLr2)) |[P∪I]| ((PEs2 ||| PEr1) |[Br]| (BLs2 ||| BLr1)))
|[B]|
(BR1 ||| BR2)
)

= (4)

hide *B∪P∪I* ***in***

(((PEs1 ||| PEr2) ||[Bs]|| (BLs1 ||| BLr2) ||[Bs]|| BR1)
||[P∪I]||
((PEs2 ||| PEr1) ||[Br]|| (BLs2 ||| BLr1) ||[Br]|| BR2)
)

With *P'* =df *P∪I*, *PR1* =df *(PEs1 ||| PEr2) ||[Bs]|| (BLs1 ||| BLr2) ||[Bs]|| BR1*, and *PR2* =df *(PEs2 ||| PEr1) ||[Br]|| (BLs2 ||| BLr1) ||[Br]|| BR2*, we may write this as:

hide *P∪B* ***in*** *(PR1 ||[P']|| PR2)*

Because of the synchronization of *PR1* and *PR2* at the hidden gates in *P'*, it is not possible to decompose the verification problem into smaller parts without assuming a further structuring of the specification parts.

Whether a further structuring can be found that is still applicable to different protocol levels (i.e., that allows the convenient definition of a variety of protocol functions) is still open to further investigation. So far, specification tasks mainly concentrated on finding an appropriate structure for the specification of a specific protocol level, not on finding a suitable 'common factor'. It is worthwhile, however, to also consider how common specification structures, supported by the use of a few specification styles, may prove advantageous in the overall design of distributed systems.