Assumption-Commitment Support for CSP Model Checking

Nick Moffat, Michael Goldsmith

Assumption-Commitment Support for CSP Model Checking

Nick Moffat[1]

Systems Assurance Group, QinetiQ
Malvern, UK

Michael Goldsmith[2]

Formal Systems (Europe) Ltd, Oxford, UK
and *Worcester College, University of Oxford*

Abstract

We present a simple formulation of Assumption-Commitment reasoning using CSP. In our formulation, an assumption-commitment style property of a process SYS takes the form $COM \sqsubseteq SYS \parallel ASS$, for some 'assumption' and 'commitment' processes ASS and COM. We state some proof rules that allow us to derive assumption-commitment style properties of a composite system from corresponding properties of its components, given appropriate side conditions. Most of the rules have a superficially appealing 'homomorphic' quality, in the sense that the overall assumption and commitment processes are composed similarly to the overall system. We also present a 'non-homomorphic' rule that corresponds quite well to proof rules of established assumption-commitment theory. The antecedants and side conditions are expressed as refinements that can be checked separately by the refinement-style model checker FDR. Examples are given to illustrate application of our theory.

Keywords: Assumption-Commitment, Assume-Guarantee, CSP, Model Checking, Compositional Reasoning

1 Introduction

The principle of *compositional program verification* is verification of a program on the basis of its constituent subprograms, without any knowledge of the interior construction of those subprograms [15]. This generalises to the notion of *compositional verification* of (hardware and/or software) systems.

Compositional verification allows large systems to be verified by reasoning *separately* about their components. So-called *compositional proof rules* are defined for program operators (more generally, for system operators). These rules take the form: "From P_1 *satisfies* ϕ_1 and ... and P_n *satisfies* ϕ_n infer P *satisfies* ϕ."

[1] Email: N.Moffat@eris.QinetiQ.com
[2] Email: michael@fsel.com

A revised version of this paper will be electronically published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

[15] Compositional verification is widely viewed as essential for verification of large systems, to counteract the state explosion problem.

We are interested in compositional reasoning when using the process algebra CSP [4,10] for modelling and reasoning about systems, especially in the context of refinement-style model checking. One form of compositional reasoning for CSP is described in [10], whereby refinement properties of a composite system can be inferred from (separately-proven) refinement properties of its components:

$$\frac{P' \sqsubseteq P \wedge Q' \sqsubseteq Q}{P' \parallel Q' \sqsubseteq P \parallel Q}$$

The rule shown is implied by monotonicity of the parallel operator $\parallel$ and transitivity of refinement. Similar rules hold for all CSP operators, since they are all monotonic.

Such rules are typically used for reasoning compositionally about systems where each component is specified independently of its environment, i.e. where the same specification would be appropriate whatever the context of the component in the wider system. However, these rules are actually powerful enough to allow compositional reasoning about more general systems: those in which components might only behave as desired in some environments. It is possible for a specification process P' to encode certain trace assumptions about the environment of a process P by arranging that P' evolves to a state in which it can exhibit any behaviours once the assumption has broken, i.e. after performing a disallowed trace. According to the terminology of [14], this amounts to assumption-commitment reasoning with an implicit assumption-commitment specification. However, this style of specification can be cumbersome. Also, it is convenient to characterise assumptions separately from commitments when clear, distinct descriptions of 'desired component behaviour' and 'supposed component environment' can be identified for a real system.

We therefore desire assumption-commitment support where assumption-commitment specifications include separate, explicit descriptions of both the environment in which components are supposed to operate correctly and the desired behaviour of the component in such an environment.

We formulate assumption-commitment properties as refinements using explicit 'assumption' and 'commitment' processes: $COM \sqsubseteq SYS \parallel ASS$. This form is suitable for checking directly using a refinement-style model checker, such as FDR [3]. However, for large systems, such properties are likely to be computationally expensive, even intractable, to check directly; hence the desire for a compositional approach.

We state some results that allow thus-formulated assumption-commitment style properties of a composite system to be deduced from corresponding properties of its components, given appropriate side conditions. For example, two theorems take the following form (with different sets of side conditions):

$$\frac{COM_1 \sqsubseteq SYS_1 \parallel ASS_1 \wedge COM_2 \sqsubseteq SYS_2 \parallel ASS_2}{COM_1 \leftrightarrow COM_2 \sqsubseteq (SYS_1 \leftrightarrow SYS_2) \parallel (ASS_1 \leftrightarrow ASS_2)}$$

The operator $\leftrightarrow$ represents 'symmetric piping': the operand processes are synchronised on shared events, which are then hidden. These theorems have a superficially appealing 'homomorphic' quality, in the sense that the overall assumption and commitment processes are composed similarly to the overall system, using the

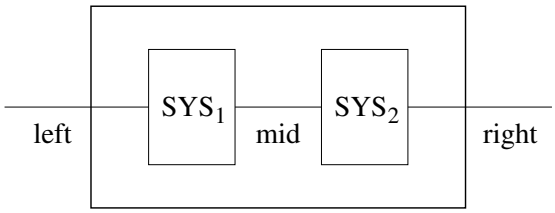

Fig. 1. An open composite system

same operator ($\leftrightarrow$, above). A 'non-homomorphic' result is also presented, which corresponds quite well to proof rules of established assumption-commitment theory [6,2]. All antecedants and side conditions are expressed as refinements that can be checked separately by refinement-style model checking. Examples are given to illustrate the results obtained.

Section 2 describes the types of composite system targetted and Section 3 explains our formulation of assumption-commitment properties as refinement properties. Section 4 presents the homomorphic theorems, and gives small examples of their use. Section 5 gives the non-homomorphic theorem, which is closer to classical assumption-commitment rules and has some advantages over the homomorphic theorems. A small example is given that illustrates its effectiveness. Space limits restrict us to proof sketches, but [7] contains detailed proofs of all the theorems. Section 6 compares our approach to related work. Section 7 gives our conclusions and outlines future work.

2 System Models

We consider a system process SYS with two components:

$$SYS = SYS_1 \underset{mid}{\parallel} SYS_2$$

where SYS_1 and SYS_2 are processes whose alphabets are contained in known alphabets $aSYS_1$ and $aSYS_2$ respectively, and $mid = aSYS_1 \cap aSYS_2$ is the synchronisation alphabet.[3] A satisfactory choice for each $aSYS_i$ would be the alphabet induced by the channels of SYS_i: all events that can be communicated on any of these channels.[4]

Putting $left = aSYS_1 \setminus aSYS_2$ and $right = aSYS_2 \setminus aSYS_1$, the system design is shown in Figure 1. For simplicity, the figure depicts *left*, *mid* and *right* as single channels, but they may correspond to multiple channels.

Sometimes the synchronisation set *mid* will be hidden, making these events internal. In such cases, the system can be modelled as follows:

$$SYS = (SYS_1 \underset{mid}{\parallel} SYS_2) \setminus mid$$

We call a system *open* if $left \cup right$ is non-empty, that is, if it has externally visible channels when the *mid* events are hidden. Otherwise, it is *closed*.

[3] Under these assumptions, $SYS = SYS_1 \ {}_{aSYS_1}\|_{aSYS_2} \ SYS_2$.

[4] We avoid using the exact alphabets of SYS_1 and SYS_2, since the exact alphabet function, usually denoted α, is not available in the machine-readable version of CSP[11].

3 Assumption-Commitment Properties

As already stated, we formulate assumption-commitment properties (AC properties) as refinements using 'assumption' and 'commitment' processes. Let SYS be a process with an associated alphabet $aSYS$ that contains all the events of SYS, and let ASS and COM be processes. For simplicity, we suppose that (like SYS) ASS and COM never act outside $aSYS$. Then we say that process SYS satisfies assumption-commitment property (ASS, COM) in CSP semantic model M if

$$(1) \quad COM \sqsubseteq_M SYS \underset{aSYS}{\parallel} ASS$$

As explained in [10], a refinement $P \sqsubseteq_M Q$ means that, in semantic model M, the behaviours of Q are behaviours of P.[5] We see that (1) can be interpreted as saying that SYS in environment ASS only exhibits behaviours 'allowed by' the commitment process COM. Monotonicity of $\parallel$ allows a stronger interpretation. It allows us to replace "in environment ASS" by "in any environment that satisfies ASS".

Example 3.1 Suppose a process SYS has channels A, B and F, so $aSYS = A \cup B \cup F$ contains all the events of SYS. (Throughout the paper, we let a channel name denote the alphabet of events communicated on that channel.) Further, suppose $ASS = RUN(A \cup B)$ and $COM = BUFF(A, B)$, where $RUN(X) = \Box\, x : X \bullet x \rightarrow RUN(X)$ is the (single state) process that can perform any sequence of events from X and never refuses an event of X and $BUFF(chan1, chan2) = chan1?x \rightarrow chan2!x \rightarrow BUFF(chan1, chan2)$ is a one place buffer from chan1 to chan2. Then the traces AC property (ASS, COM) of SYS expresses that all traces of SYS are traces of a buffer from A to B as long as the environment of SYS never performs the F event. Here, F may be a failure event that causes SYS to fail. The assumption in this case has a very simple form in that it allows a subset of events in its alphabet, regardless of earlier system activity. All AC properties considered in this paper have similarly simple assumptions, though more complex assumptions are expressible. □

This explicit formulation of AC properties is suitable for checking directly using a refinement-style model checker, such as FDR[3]. But this can be computationally expensive for large systems. A compositional approach is possible using the results in Sections 4 and 5.

Classically [6,2], assumption-commitment theory is set in the context of concurrent systems built of sequential components that interact by message passing. Systems are built from sub-systems (ultimately from sequential components) by parallel and sequential composition. A *history variable* records the communication history. Each component's local state is held in its *program variables*. *Logical variables* remember the values of input variables in the initial state. Classic AC properties are expressed as *assumption-commitment correctness formulae* of the form:

[5] We consider CSP's semantic models T, F and N. In the traces model T, a process's denotational value is the prefix-closed set of all finite traces that it can perform. The stable failures model F additionally records the failures (t, X) of a process, where t is a trace to some stable state (a state in which no internal activity is possible) and X is a refusal in that state (a set of events all of which can be refused simultaneously). The failures-divergences model records the failures (all of them, not just the stable ones) and the divergences, which are the traces to a divergent state (one in which an infinite sequence of internal actions if possible). We use 'behaviour' to mean a trace, a stable or unstable failure, or a divergence, as appropriate for the semantic model. Further details can be found in [10].

(2) $\langle A, C\rangle : \{\phi\}\, P\, \{\psi\}$

where A and C are assumption and commitment predicates over the communication history and logical variables, and ϕ and ψ are pre-condition and post-condition predicates over communication history, logical variables and program variables.

In essence, a valid AC formula for a component P is interpreted in [2] as follows: if ϕ holds initially (i.e. in the state in which P starts its execution) then C holds initially and, provided also that A holds after all preceding communications, C holds after every communication and ψ holds on termination (if this occurs).

However, our formulation omits the pre- and post-conditions, since we are currently only concerned with state insofar as it is manifested by process communications and we are focussed on characterising unparameterised systems composed using parallel process operators alone. [6]

How closely does our formulation mirror the assumption and commitment aspects of the classical theory? We address this question informally for the traces model. Recall our informal interpretation of equation (1): SYS in any environment that satisfies ASS only exhibits behaviours 'allowed by' COM. Let o be an output event of SYS. For any trace t, $t^\frown\langle o\rangle$ denotes t extended by the single event o. Then, for all traces $t^\frown\langle o\rangle$ of SYS in an environment that satisfies ASS, $t^\frown\langle o\rangle$ is a trace of COM. That is, for all traces $t^\frown\langle o\rangle$ of SYS, if $t^\frown\langle o\rangle$ is a trace of ASS then it is a trace of COM. A condition we impose later (liberality of assumption processes on component output events) then gives: for all traces $t^\frown\langle o\rangle$ of SYS, if t is a trace of ASS then $t^\frown\langle o\rangle$ is a trace of COM. This amounts to an output of SYS being allowed by COM if the trace up to the current state is a trace of ASS, which corresponds well to the classical interpretation of AC formulae.

We believe that expression of assumption and commitment as processes is novel and interesting. Alternative formulations using CSP processes are possible, but they are not considered here.

4 Homomorphic Theorems

The theorems in this section find an assumption-commitment property of a given SYS. They require that the alphabets $aSYS_1$ and $aSYS_2$ are defined and meet the conditions described in Sections 2 and 3. Formally, they require that $\alpha(SYS_i) \cup \alpha(ASS_i) \cup \alpha(COM_i) \subseteq aSYS_i$, $(i = 1, 2)$, where the function α gives the exact alphabet of a process. As shorthand, we say that $aSYS_1$ and $aSYS_2$ are *healthy*.

The first theorem applies when SYS_1 and SYS_2 are composed by synchronising on shared events and leaving these visible:

Theorem 4.1 (Shared events visible, mutual dependence permitted) *For*

[6] Our approach could be extended to consider AC properties of parameterised systems, in which case parameters would be considered as contributing to the initial state. We also expect that other process operators could be handled, in particular sequential composition.

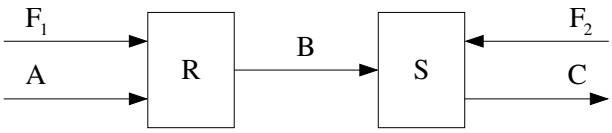

Fig. 2. A faulty buffer open system. R and S behave (in the traces model) as buffers from A to B and from B to C while no signals have occurred on channels F_1 and F_2, respectively.

$M = T,\ F \text{ or } N,$

$$\frac{COM_1 \sqsubseteq_M SYS_1 \underset{aSYS_1}{\parallel} ASS_1 \qquad COM_2 \sqsubseteq_M SYS_2 \underset{aSYS_2}{\parallel} ASS_2}{COM_1 \underset{mid}{\parallel} COM_2 \sqsubseteq_M (SYS_1 \underset{mid}{\parallel} SYS_2) \underset{aSYS}{\parallel} (ASS_1 \underset{mid}{\parallel} ASS_2)}$$

where $aSYS_1$, $aSYS_2$ *are healthy,* $mid = aSYS_1 \cap aSYS_2$ *and* $aSYS = aSYS_1 \cup aSYS_2$.

The conditions in the antecedant are simply the individual AC properties of the components. The consequent is easily derived algebraicly. Monotonicity gives $COM_1 \underset{mid}{\parallel} COM_2 \sqsubseteq_M (SYS_1 \underset{aSYS_1}{\parallel} ASS_1) \underset{mid}{\parallel} (SYS_2 \underset{aSYS_2}{\parallel} ASS_2)$. The consequent is then obtained by repeated application, to the right-hand side and its sub-formulae, of the laws $X \parallel Y$-assoc, $X \parallel Y$-sym and equivalence of $P \ {}_X\|_Y \ Q$ and $P \underset{X \cap Y}{\parallel} Q$ when $\alpha(P) \subseteq X$ and $\alpha(Q) \subseteq Y$, all stated in [10].

Example 4.2 Figure 2 depicts a system with two components and five channels. Component R inputs values on A and F_1, and outputs values on B; component S inputs values on B and F_2, and outputs values on C. We suppose that channels A, B and C all carry the same type of data. The system of Figure 2 can be expressed in CSP as $SYS = R \underset{B}{\parallel} S$.

Suppose R and S satisfy AC properties (ASS_R, COM_R) and (ASS_S, COM_S) in CSP's traces model, for assumption and commitment processes as follows:

$$ASS_R = RUN(A \cup B) \qquad ASS_S = RUN(B \cup C)$$
$$COM_R = BUFF(A, B) \qquad COM_S = BUFF(B, C)$$

for alphabets $aSYS_R$ and $aSYS_S$ defined as follows:

$$aSYS_R = A \cup B \cup F_1 \qquad aSYS_S = B \cup C \cup F_2$$

These individual AC properties are similar to the AC property discussed in Example 3.1. Each expresses that the component acts like a one place buffer from an input to an output, as long as no events occur on respective failure channel F_1 or F_2.

We seek overall assumption and commitment processes ASS and COM such that the composite system SYS satisfies AC property (ASS, COM) in the traces model, i.e. such that $COM \sqsubseteq_T SYS \underset{aSYS}{\parallel} ASS$, for a healthy $aSYS$.

Putting $mid = aSYS_R \cap aSYS_S = B$, Theorem 4.1 is applicable. Its assumptions are satisfied, including the alphabet conditions. Recalling that $SYS = R \underset{B}{\parallel} S$, we deduce that $COM \sqsubseteq_T SYS \underset{aSYS}{\parallel} ASS$, where

$$ASS = ASS_R \underset{B}{\parallel} ASS_S \qquad COM = COM_R \underset{B}{\parallel} COM_S$$
$$= RUN(A \cup B) \underset{B}{\parallel} RUN(B \cup C) \qquad = BUFF(A, B) \underset{B}{\parallel} BUFF(B, C)$$
$$= RUN(A \cup B \cup C)$$
$$aSYS = aSYS_R \cup aSYS_S = A \cup B \cup C \cup F_1 \cup F_2$$

The deduced assumption-commitment property expresses that, while no signals have occurred on either fault channel, the traces of the composed system are traces of a two place buffer. □

The usefulness of Theorem 4.1 is limited because the resulting assumption process synchronises with SYS on all events and the constraints on such events made by each individual assumption are preserved in the overall assumption. If the individual assumptions constrain *mid* events, these constraints must not render the overall assumption useless. For example, *mid* events might properly be considered outside the control of a realistic environment, in which case no environment for SYS could be implemented to guarantee that the overall assumption is met.

On the other hand, it may be that constraints on *mid* events can be implemented by some monitoring and control mechanisms not modelled. Also, it is possible that the aim is merely to characterise system 'reliability', which could be done by obtaining an overall AC property such that the assumption constrains *mid* events (even if outside environmental control) and then appealing to some characterisation of the likelihood of this assumption being met.

Theorems 4.3 and 4.5, both given below, apply when SYS_1 and SYS_2 are composed by symmetric piping $\leftrightarrow$, i.e. by synchronising on shared events (*mid*) and then hiding them. Because *mid* events are hidden, there is no opportunity for a resulting assumption to constrain them directly.

Some extra terminology is needed at this point. First, we recall the notions of lazy abstraction and separability described in [10]. In CSP's traces model lazy abstraction is equivalent to hiding. In the richer models it is like hiding except that it avoids introducing (operational) divergence. The examples in this paper all use the traces model, so the distinction can be ignored for these.

Let $LAbs_X(ASS)$ denote the process obtained by lazily abstracting ASS away from alphabet X. Let $LProj_X(ASS)$ denote $LAbs_{\Sigma \setminus X}(ASS)$, the process obtained by abstracting away from all events outside X, which we call the lazy projection of ASS to X.

An assumption process ASS is

- *separable on X in M* if $ASS =_M LAbs_X(ASS) \;|||\; LProj_X(ASS)$;
- *neutral on X in M* if $ASS =_M LAbs_X(ASS) \;|||\; RUN(X)$.

Both conditions can be checked using FDR. Separability is quite a strong property, corresponding to complete independence of behaviour w.r.t. two sets of events: X and its complement. Neutrality is even stronger than separability. To be neutral on a set of events X in the traces model T is to be separable on X in T and capable of performing any sequence of events from X. (Extra properties are implied by neutrality in the richer semantic models F and N.)

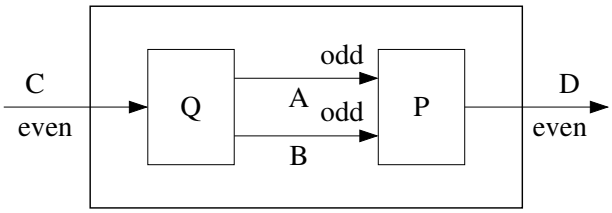

Fig. 3. A simple pipeline open system, with hidden channels

The remaining theorems of this section require separability, or neutrality, on *mid*. Theorem 4.3 yields an assumption that does not constrain internal events but it imposes separability and neutrality side conditions.

Theorem 4.3 (Shared events hidden, no mutual dependence) *For $M = T$ or F,*

$$\frac{\begin{array}{c} COM_1 \sqsubseteq_M SYS_1 \underset{aSYS_1}{\parallel} ASS_1 \quad COM_2 \sqsubseteq_M SYS_2 \underset{aSYS_2}{\parallel} ASS_2 \\ LProj_{mid}(ASS_1) \sqsubseteq_M LProj_{mid}(COM_2) \quad LProj_{mid}(ASS_2) \sqsubseteq_M LProj_{mid}(COM_1) \\ ASS_1 \text{ is neutral on } mid \text{ in } M \quad ASS_2 \text{ is separable on } mid \text{ in } M \end{array}}{COM_1 \leftrightarrow COM_2 \sqsubseteq_M (SYS_1 \leftrightarrow SYS_2) \underset{aSYS}{\parallel} (ASS_1 \leftrightarrow ASS_2)}$$

where $aSYS_1$, $aSYS_2$ are healthy, $mid = aSYS_1 \cap aSYS_2$, $aSYS = (aSYS_1 \cup aSYS_2) \setminus mid$ and $\cdot \leftrightarrow \cdot = (\cdot \underset{mid}{\parallel} \cdot) \setminus mid$.

In addition to the individual AC properties, there are two conditions of the form $LProj_{mid}(ASS_i) \sqsubseteq_M LProj_{mid}(COM_j)$. These state that the commitment process for component j projected onto the interface alphabet *mid* must refine the assumption process for component i projected onto the same alphabet. Essentially, these conditions say that each component's assumption on *mid* is satisfied by the other's commitment on *mid* – a natural condition to impose in this context.

Unfortunately, the proof is long and difficult to sketch. Briefly, using montonicity of CSP, neutrality of SYS_1 on *mid*, that $RUN(X)$ is unit for $\underset{X}{\parallel}$ in all models M, and that disjoint interleaving then shared parallel is equivalent to successive shared parallels on separate partitions, we obtain that $COM_1 \leftrightarrow COM_2 \sqsubseteq_M ((SYS_1 \underset{mid}{\parallel} SYS_2) \underset{aSYS}{\parallel} (LAbs_{mid}(ASS_1) \,|||\, LAbs_{mid}(ASS_2))) \setminus mid$. This can then be reduced to the consequent, using separability of SYS_1 on *mid* and separability of SYS_2 on *mid*.

This theorem requires separability of ASS_2 on *mid* and the stronger property of neutrality of ASS_1 on *mid*. One way this can occur is in a unidirectional pipeline, with *mid* containing only outputs of SYS_1 and inputs of SYS_2.

Example 4.4 Consider the pipeline system depicted in Figure 3. It consists of two components, two external channels, and two internal channels. Q inputs values on C and outputs values on A and B; P inputs values on A and B, and outputs values on D. It can be expressed in CSP as $SYS = (Q \underset{A \cup B}{\parallel} P) \setminus A \cup B$.

Suppose Q and P satisfy AC properties (ASS_Q, COM_Q) and (ASS_P, COM_P) in CSP's traces model, for assumption and commitment processes as follows:

$$ASS_Q = RUN(A \cup B \cup EvenC) \qquad ASS_P = RUN(OddA \cup OddB \cup D)$$

$$COM_Q = RUN(OddA \cup OddB \cup C) \quad COM_P = RUN(A \cup B \cup EvenD)$$

where the alphabets have the suggested meanings and for alphabets $aSYS_Q$ and $aSYS_P$ defined as follows:

$$aSYS_Q = A \cup B \cup C \qquad aSYS_P = A \cup B \cup D$$

We seek assumption and commitment processes ASS and COM such that the composite system SYS satisfies AC property (ASS, COM) in the traces model, i.e. such that $COM \sqsubseteq_T SYS \underset{aSYS}{\parallel} ASS$, for some healthy $aSYS$.

Putting $mid = aSYS_Q \cap aSYS_P = A \cup B$, Theorem 4.3 is applicable. Its assumptions are satisfied, including the alphabet conditions. In particular, ASS_Q is neutral on mid. Recalling that $SYS = (Q \underset{A \cup B}{\parallel} P) \setminus A \cup B$, we deduce that $COM \sqsubseteq_T SYS \underset{aSYS}{\parallel} ASS$, where

$$\begin{aligned}
ASS &= ASS_Q \leftrightarrow ASS_P \\
&= (RUN(A \cup B \cup EvenC) \underset{A \cup B}{\parallel} RUN(OddA \cup OddB \cup D)) \setminus A \cup B \\
&= RUN(EvenC \cup D) \\
COM &= COM_Q \leftrightarrow COM_P \\
&= (RUN(OddA \cup OddB \cup C) \underset{A \cup B}{\parallel} RUN(A \cup B \cup EvenD)) \setminus A \cup B \\
&= RUN(C \cup EvenD) \\
aSYS &= (aSYS_Q \cup aSYS_P) \setminus mid = C \cup D
\end{aligned}$$

The deduced assumption-commitment property expresses that all outputs of SYS on D are even if all (previous) inputs of SYS on C are even. □

Many non-pipelined systems are also covered by this theorem, since it allows data to flow in both directions: neutrality of ASS_1 (which is merely an assumption process) requires only that it does not constrain the shared interface events. Even if values are communicated in both directions between the components, it may be that the promise that SYS_1 will meet its commitment is not conditional on behaviour at the shared interface.

Unfortunately, Theorem 4.3 does not apply if each component's assumption constrains behaviour at the shared interface – no mutual dependence is permitted.

Theorem 4.5 weakens one of the conditions to admit systems with mutual assumptions: it requires separability of each component assumption process, instead of separability of one and neutrality of the other.

The key challenge posed by mutual dependence is the avoidance of circular reasoning. Theorems 4.5 and 5.1 avoid circular reasoning by imposing extra conditions to ensure that assumptions cannot break 'simultaneously', i.e. by the occurrence of a single communication event. In particular, these theorems place extra 'liberality' conditions (defined below) on the assumption processes.

We say a process P is *liberal on* X if $P \mid\mid\mid RUN(\Sigma \setminus X) \sqsubseteq_T RUN(\Sigma)$, where Σ

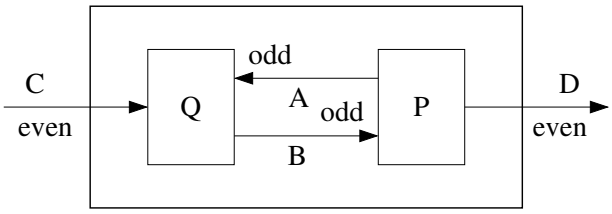

Fig. 4. An open system with mutually dependent components and hidden channels

denotes the set of all defined events. This condition amounts to P never being able to refuse any events of X. As with the other defined conditions, liberality on a set can be checked using FDR.

CSP channels are directionless – any process may input values to a particular channel (and so bind a variable) or output values to it. However, liberality will be required on defined sets of 'input' or 'output' events of a process. The theorems below hold for any partitioning of component events into inputs and outputs, but they are most likely to be useful when the CSP channels are considered to have the same 'directions' as the system communications modelled.

For the liberality conditions to prevent simultaneous breakage of the component assumptions, the theorems below rely on a further condition: that each shared event is an input of one component process and an output of the other. We call this *in/out synchrony.*[7] It is a non-trivial condition as CSP is capable of modelling more complex 'plumbing' of processes, but it accords with the usual model of synchrony in classical assumption-commitment theory [2,6].

Theorem 4.5 (Shared events hidden, mutual dependence permitted) *For* $M = T$ *or* F,

$$\frac{\begin{array}{rr} COM_1 \sqsubseteq_M SYS_1 \underset{aSYS_1}{\parallel} ASS_1 & COM_2 \sqsubseteq_M SYS_2 \underset{aSYS_2}{\parallel} ASS_2 \\ LProj_{mid}(ASS_1) \sqsubseteq_M LProj_{mid}(COM_2) & LProj_{mid}(ASS_2) \sqsubseteq_M LProj_{mid}(COM_1) \\ ASS_1 \text{ is separable on } mid \text{ in } M & ASS_2 \text{ is separable on } mid \text{ in } M \\ ASS_1 \text{ is liberal on } SYS_1^{Outs} \cap mid & ASS_2 \text{ is liberal on } SYS_2^{Outs} \cap mid \end{array}}{COM_1 \leftrightarrow COM_2 \sqsubseteq_M (SYS_1 \leftrightarrow SYS_2) \underset{aSYS}{\parallel} (ASS_1 \leftrightarrow ASS_2)}$$

where $aSYS_1$, $aSYS_2$ *are healthy,* $mid = aSYS_1 \cap aSYS_2$, $aSYS = (aSYS_1 \cup aSYS_2) \setminus mid$, $\cdot \leftrightarrow \cdot = (\cdot \underset{mid}{\parallel} \cdot) \setminus mid$, *and* SYS_1 *and* SYS_2 *are in/out synchronous.*

The proof is similar to that for Theorem 4.3. It differs in that the initial reduction step is more complex. Central to this reduction is a lemma that says $(P \underset{X}{\parallel} Q) \underset{X}{\parallel} (R \underset{X}{\parallel} S) \sqsubseteq_M P \underset{X}{\parallel} R$ when $Q \sqsubseteq_M LProj_X(R \underset{X}{\parallel} S)$, $S \sqsubseteq_M LProj_X(P \underset{X}{\parallel} Q)$, and $\forall\, t \in \text{traces}(Q) \cap \text{traces}(S)$, $X \cap \text{Union}(\text{refusals}(Q/t) \cap \text{refusals}(S/t)) = \varnothing$. [7] contains the details.

[7] Note that in/out synchrony is not needed by Theorems 4.1 and 4.3.

Example 4.6 Consider the system depicted in Figure 4. Q inputs values on channels A and C, and outputs values on channel B; P inputs values on B and outputs values on A and D. It can be obtained by reversing channel A in Example 4.4.

Suppose Q and P satisfy AC properties (ASS_Q, COM_Q) and (ASS_P, COM_P) in CSP's traces model, for assumption and commitment processes as follows:

$$ASS_Q = RUN(OddA \cup B \cup EvenC) \quad ASS_P = RUN(A \cup OddB \cup D)$$

$$COM_Q = RUN(A \cup OddB \cup C) \quad COM_P = RUN(OddA \cup B \cup EvenD)$$

where the alphabets have the suggested meanings and $aSYS_Q$ and $aSYS_P$ are defined as follows:

$$aSYS_Q = A \cup B \cup C \quad aSYS_P = A \cup B \cup D$$

We seek assumption and commitment processes ASS and COM such that the composite system

$$SYS = Q \underset{A \cup B}{\parallel} P$$

satisfies AC property (ASS, COM) in T, that is, such that $COM \sqsubseteq_T SYS \underset{aSYS}{\parallel} ASS$, for some healthy $aSYS$.

Putting $mid = aSYS_Q \cap aSYS_P = A \cup B$, Theorem 4.5 is applicable. Its assumptions are satisfied, including all the side conditions.[8] We deduce that $COM \sqsubseteq_T SYS \underset{aSYS}{\parallel} ASS$ for this SYS, where

$$\begin{aligned}
ASS &= ASS_Q \leftrightarrow ASS_P \\
&= (RUN(OddA \cup B \cup EvenC) \underset{mid}{\parallel} RUN(A \cup OddB \cup D)) \setminus A \cup B \\
&= RUN(EvenC \cup D) \\
COM &= COM_Q \leftrightarrow COM_P \\
&= (RUN(A \cup OddB \cup C) \underset{mid}{\parallel} RUN(OddA \cup B \cup EvenD)) \setminus A \cup B \\
&= RUN(C \cup EvenD) \\
aSYS &= (aSYS_Q \cup aSYS_P) \setminus mid = C \cup D
\end{aligned}$$

The deduced AC property expresses that all outputs on channel D are even if all (previous) inputs on channel C are even. □

The next section gives a more useful result, with an example of its application. It will be shown there that none of the homomorphic theorems yield a useful result for that example.

5 A More Useful Theorem

The following theorem corresponds quite closely to the classical proof rules for parallel composition in the assumption-commitment literature [6,2]. We expect it to be more useful than the homomorphic theorems of Section 4, since it does not require the individual assumptions to be neutral, or even just separable, on the shared events.

[8] Note that Theorem 4.3 is *not* applicable; neither ASS_Q nor ASS_P is neutral on mid. However, each is separable and satisfies the liberality conditions of Theorem 4.5.

Theorem 5.1 (Shared events visible, mutual dependence permitted, assumption given) *For* $M = T$,

$$\frac{\begin{array}{cc} COM_1 \sqsubseteq_M SYS_1 \underset{aSYS_1}{\parallel} ASS_1 & COM_2 \sqsubseteq_M SYS_2 \underset{aSYS_2}{\parallel} ASS_2 \\ ASS_1 \text{ is liberal on } SYS_1^{Outs} \cap mid & ASS_2 \text{ is liberal on } SYS_2^{Outs} \cap mid \\ COM_1 \text{ is liberal on } SYS_1^{Ins} \cap mid & COM_2 \text{ is liberal on } SYS_2^{Ins} \cap mid \\ ASS_1 \sqsubseteq_M LProj_{aSYS_1}(ASS \underset{aSYS_2}{\parallel} COM_2) & ASS_2 \sqsubseteq_M LProj_{aSYS_2}(ASS \underset{aSYS_1}{\parallel} COM_1) \end{array}}{COM_1 \underset{mid}{\parallel} COM_2 \sqsubseteq_M (SYS_1 \underset{mid}{\parallel} SYS_2) \underset{aSYS}{\parallel} ASS}$$

where $aSYS_1$, $aSYS_2$ *are healthy,* $mid = aSYS_1 \cap aSYS_2$, $aSYS = aSYS_1 \cup aSYS_2$, *and* SYS_1 *and* SYS_2 *are in/out synchronous.*

As usual, the first two conditions in the antecedant are simply the individual AC properties. The next four conditions are natural liberality constraints on the individual ASS_i and COM_i processes. Essentially, each ASS_i liberality condition states that satisfaction of ASS_i cannot be invalidated by outputs from SYS_i to SYS_j. Conversely, each COM_i liberality condition states that satisfaction of COM_i cannot be invalidated by inputs to SYS_i from SYS_j. We expect that these conditions will frequently be satisfied, since the use of non-liberal ASS_i and COM_i processes would be inappropriate when all messages are output by at most one component and components cannot refuse inputs. [9]

The last two conditions involve a process ASS, which is the overall assumption process in the consequent. These conditions state that each individual assumption ASS_i is refined by the following process: ASS synchronised with the other commitment COM_j on the alphabet $aSYS_j$ and then lazily projected to the alphabet $aSYS_i$. These may appear the least natural of the conditions, but they correspond to natural conditions in the classical AC theory of the form 'Overall Assumption' $\wedge$ 'Commitment for component j' $\Rightarrow$ 'Assumption for component i'. They ensure that each individual component assumption on the shared interface is enforced by the commitment of the other component.

[7] contains a detailed soundness proof. It uses induction on trace length and considers two separate cases: failure of one component to meet its commitment before the other has failed, and failure of both components on the same event. The first case is impossible because the non-failed commitment and the overall assumption ASS together satisfy the assumption of the 'failed' system, so its commitment must also be satisfied. Impossibility of the second case is argued using in/out synchrony and liberality: the common event must be an input of one component and an output of the other, and liberality carries satisfaction of assumptions and commitments before this event over to satisfaction after this event.

Importantly, the overall assumption ASS only appears in the last two conditions of the antecedant. Since these conditions ensure that the assumptions on the shared interface are enforced by the commitments of SYS_1 and SYS_2, ASS need only enforce

[9] There would be little point having an assumption that restricts the outputs of a process, since it could be weakened to one that does not, without affecting satisfaction of any commitment. Similarly, no commitment that restricts the inputs of a process could form part of a valid AC property in this concurrency model.

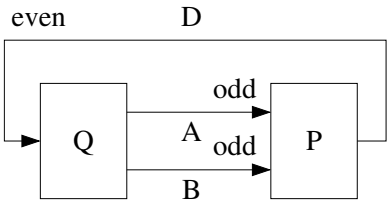

Fig. 5. A closed system with mutually dependent components

any remaining aspects of the component assumptions (which will be on the external interfaces of SYS_1 and SYS_2). In the following example, no such restrictions remain, so ASS can be $RUN(aSYS)$.

Example 5.2 This example is based on one described in [2], itself due to [13].

Consider the system depicted in Figure 5, which can be obtained by connecting channels C and D in Example 4.4, thus introducing a feedback loop (channel D; the name C is dropped).

Suppose that Q and P satisfy assumption-commitment properties (ASS_Q, COM_Q) and (ASS_P, COM_P) in CSP's traces model, for the following assumption and commitment processes:

$$ASS_Q = RUN(A \cup B \cup EvenD) \qquad ASS_P = RUN(OddA \cup OddB \cup D)$$

$$COM_Q = RUN(OddA \cup OddB \cup D) \quad COM_P = RUN(A \cup B \cup EvenD)$$

where the alphabets have the suggested meanings and $aSYS_Q$ and $aSYS_P$ are defined as follows:

$$aSYS_Q = A \cup B \cup D \qquad aSYS_P = A \cup B \cup D$$

Now suppose we wish to obtain a commitment for the composite system, but in this case without assuming anything of the environment. This leads us to express the desired top-level assumption process:

$$ASS = RUN(A \cup B \cup D)$$

When put in parallel with a system process on shared alphabet $A \cup B \cup D$, ASS has no constraining effect in the traces semantic model; it therefore represents the assumption *true* in the traces model when used as the assumption process.

We seek a commitment process COM such that the composite system

$$SYS = Q \underset{A \cup B \cup D}{\parallel} P$$

satisfies AC property (ASS, COM) in the traces model, i.e. such that $COM \sqsubseteq_T SYS \underset{aSYS}{\parallel} ASS$, for some healthy $aSYS$.

Theorem 5.1 is applicable. Its assumptions are satisfied, including all the side conditions. We deduce that $COM \sqsubseteq_T SYS \underset{aSYS}{\parallel} ASS$, where

$$\begin{aligned} COM &= COM_Q \underset{mid}{\parallel} COM_P \\ &= RUN(OddA \cup OddB \cup D) \underset{mid}{\parallel} RUN(A \cup B \cup EvenD) \\ &= RUN(OddA \cup OddB \cup EvenD) \end{aligned}$$

$$aSYS = aSYS_Q \cup aSYS_P = A \cup B \cup D$$

The deduced assumption-commitment property expresses that all values communicated on channel D are even and all those communicated on A and B are odd, under the trivial assumption. □

The applicable homomorphic theorems for this example are Theorems 4.1 and 4.5. However, neither of these yields useful results for this particular example. The former theorem gives $RUN(OddA \cup OddB \cup EvenC) \sqsubseteq SYS \underset{A \cup B \cup D}{\parallel} RUN(OddA \cup OddB \cup EvenC)$, which amounts to saying that SYS has the desired behaviour, on the assumption that it has the desired behaviour. The latter theorem gives $STOP \sqsubseteq STOP$, which again is not useful. (Nevertheless, the homomorphic theorems appear to be useful for some open systems, as demonstrated by Examples 4.2, 4.4 and 4.6).

6 Related Work

There is much work in the area of assumption-commitment reasoning. We have mentioned [6,15,2] and in Section 3 we compared our formulation of assumption-commitment properties to the classical approach.

Pandya [8] established an assumption-commitment verification style for CSP programs using first order logic assertions over finite traces. Similarly, Kay and Reed present deductive rules in [5] for compositional reasoning about CSP processes. Both approaches use predicates, rather than assumption and commitment processes. They are suited to assumption-commitment reasoning about CSP processes using theorem proving rather than model checking. Another difference is that these approaches are restricted to the traces model; some of our results additionally hold in F and N, though their utility is unclear for these richer models.

Evans, Treharne and Schneider have a decomposition rule, for their $CSP \parallel B$ architecture, that allows rely/guarantee reasoning to establish consistency results for composite systems [12].

Zhou Chaochen's notion of 'weakest environment' [1] is the concurrent analogue of weakest pre-condition: it is the weakest environment in which a given process (SYS) satisfies a given property (refinement of COM). It would be interesting to investigate the opportunity for deriving a weakest assumption ASS, rather than testing whether Theorem 5.1 applies for a given ASS.

[9] compares some implementations of assume-guarantee approaches using the SPIN and SMV model checkers, which use temporal logic specifications.

7 Conclusions and Future Work

We have presented a simple formulation of assumption-commitment properties for CSP using refinement, and some theorems that allow such assumption-commitment properties of composite systems to be deduced from separately provable properties of their components. All side conditions and conditions appearing in the antecedants are expressible as refinements in a form suitable for checking using FDR. Moreover, no such condition involves more than one SYS_i component.

The 'homomorphic' theorems appear to be quite useful, but they have some

limitations:

- In the $\|$ case, the resulting assumption ASS is excessively strong in that it can restrict shared events, which makes it unenforceable in realistic environments when these events should properly be considered internal. Theorem 4.1 failed to full take advantage of the commitments, which would have allowed a weaker overall assumption.
- In the $\leftrightarrow$ cases, the resulting assumptions ASS are excessively weak, requiring stringent side conditions to compensate for loss of correspondence between ASS, SYS and COM on mid events.

Theorem 5.1 comes quite close to the established assumption-commitment theory, expressed in the context of CSP model checking. Future work will investigate its application to large examples.

The current theory supports compositional reasoning about systems that can be modelled as a shared parallel composition of two component processes. We believe this is a significant step towards effective compositional reasoning using refinement-style model checking of CSP, because our experience is that many systems can be modelled in this way and that where state explosion occurs it tends to arise from parallel composition of processes. Further, classical AC reasoning has found application even though it is focussed on only simple forms of parallel and sequential composition. Even so, it may be worth extending the current theory to composition over other CSP process operators, including sequential composition; in this case it would seem sensible to include pre- and post-conditions to address issues of state.

It appears likely that the current restriction to binary shared parallel composition could be lifted. The same intuition that underlies the current theory can be applied to systems with many components composed in replicated alphabetised parallel (which synchronises each component process on its interface to the other components). Meanwhile, multiple-component systems can be reasoned about by successive application of the current theory to two-component systems, in a hierarchical fashion.

Future work will develop guidance for the expression of AC properties using assumption and commitment processes, which is quite different to their classical expression as predicates over a history variable and logical variables.

8 Acknowledgements

Early discussions with William Simmonds lead to a similar formulation of AC properties. We received helpful early comments from Cliff Jones. We also thank the reviewers for their useful comments.

References

[1] Chaochen, Z., *Weakest environment of communicating processes*, in: *Proceedings of NCC '82, Houston*, 1982.

[2] de Roever, W.-P., F. de Boer, U. Hannemann, J. Hooman, Y. Lakhnech, M. Poel and J. Zwiers, "Concurrency Verification: Introduction to Compositional and Noncompositional Methods," Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, 2001.

[3] Formal Systems (Europe) Ltd, "Failures-Divergence Refinement: FDR 2 User Manual," (1992-2006).

[4] Hoare, C. A. R., *Communicating sequential processes*, Communications of the ACM **21** (1978), pp. 666–677.

[5] Kay, A. and J. N. Reed, *A rely and guarantee method for timed CSP: A specification and design of a telephone exchange*, IEEE Trans. Software Eng. **19** (1993), pp. 625–639.

[6] Misra, J. and K. M. Chandy, *Proofs of networks of processes*, IEEE Transactions on Software Engineering **7** (1981), pp. 417–426.

[7] Moffat, N. and M. Goldsmith, *Assumption-commitment via CSP*, Technical report QINETIQ/S&DU/TIM/TR0601826, QinetiQ (2006), available from the authors.

[8] Pandya, P. K., *Some comments on the assumption-commitment framework for compositional verification of distributed programs.*, in: J. W. de Bakker, W. P. de Roever and G. Rozenberg, editors, *REX Workshop*, Lecture Notes in Computer Science **430** (1989), pp. 622–640.

[9] Pasareanu, C., M. B. Dwyer and M. Huth, *Assume-Guarantee Model Checking of Software: A Comparative Case Study*, in: *6th International SPIN Workshop on Practical Aspects of Model Checking*, LNCS **1680** (1999), pp. 168–183.
URL http://pubs.doc.ic.ac.uk/model-checking-stubs/

[10] Roscoe, A. W., "The Theory and Practice of Concurrency," Prentice Hall, 1998, ISBN 0-13-6774409-5, pp. xv+565.

[11] Scattergood, J. B., "Tools for CSP and Timed CSP," Ph.D. thesis, Oxford University Computing Laboratory (1998).

[12] Schneider, S., H. Treharne and N. Evans, *Chunks: Component verification in CSP parallel B*, Lecture Notes in Computer Science **3771** (2005).

[13] Shankar, N., *Lazy compositional verification*, in: W.-P. de Roever, H. Langmaack and A. Pnueli, editors, *Compositionality: The Significant Difference, Proceedings of the International Symposium COMPOS '97, Malente, Germany, September 7-12, 1997*, LNCS **1536** (1998), pp. 541–564.

[14] Stølen, K., F. Dederichs and R. Weber, *Specification and refinement of networks of asynchronously communicating agents using the assumption / commitment paradigm*, Formal Aspects of Computing **3** (1995).

[15] Zwiers, J., "Compositionality and Partial Correctness," LNCS **321**, Springer-Verlag, 1989.