Model Checking under Fairness in ProB and its Application to Fair Exchange Protocols

David M. Williams[1], Joeri de Ruiter[2], and Wan Fokkink[1,3]

[1] Department of Computer Science, VU University Amsterdam
[2] Institute for Computing and Information Science, Radboud University Nijmegen
[3] Faculty of Mechanical Engineering, Eindhoven University of Technology

Abstract. Motivated by Murray's work on the limits of refinement testing for CSP, we propose the use of ProB to check liveness properties under assumptions of strong and weak event fairness, whose refinement-closures cannot generally be expressed as refinement checks for FDR. Such properties are necessary for the analysis of fair exchange protocols in CSP, which assume at least some messages are sent over a resilient channel. As the properties we check are refinement-closed, we retain CSP's theory of refinement, enabling subsequent step-wise refinement of the CSP model. Moreover, we improve upon existing CSP models of fair exchange protocols by proposing a revised intruder model inspired by the one of Cederquist and Dashti. Our intruder model is stronger as we use a weaker fairness constraint.

1 Introduction

Hoare's Communicating Sequential Processes (CSP) [1] is a process algebra for describing models of interacting processes in terms of the events that they perform. For two decades the Failures Divergence Refinement (FDR) checker [2] has been the principal tool for verifying properties of models expressed in CSP. FDR tests whether the CSP model of the system being analysed refines some specification of the system's desired behaviour, which is also written in CSP. This differentiates FDR from other model checkers which test whether a system satisfies some predicate expressed in a temporal logic.

In [3], Lowe investigated the extent to which it can be checked that CSP processes satisfy temporal logic specifications using a refinement-based model checker, such as FDR. He defined the atomic formulae of a temporal logic he considered appropriate for specifying communicating processes. Subsequently, as a result of his investigation into the limits of refinement testing for CSP, Murray concluded that alternative verification approaches besides refinement checking for CSP should be further pursued [4]. Murray demonstrated there exist useful predicates that cannot generally be expressed as refinement checks in any semantic model of CSP that FDR can handle. One such class of predicates includes liveness properties under Murray's refinement-closed notions of strong and weak event fairness.

We demonstrate how to directly check whether CSP processes satisfy predicates expressed in Lowe's temporal logic using ProB, which is a tool that facilitates LTL model checking for a number of formalisms including CSP [5]. Not all of the operators offered by ProB for LTL model checking match the intended meaning of their counterpart in the grammar defined by Lowe. However, we shall show how one can express Lowe's temporal logic in ProB. By using ProB to check that a CSP process P satisfies a formula S, written in Lowe's grammar, we can be sure that checking $P \sqsubseteq_{\mathcal{RT}} P'$ in FDR guarantees that $P' \models S$. This is necessary when $P \models S$ cannot be expressed as a simple refinement check $\mathit{Spec}(S) \sqsubseteq P$ in $\mathcal{RT}$ or any other semantic model $\mathcal{M}$ that FDR can handle.

Model checking fair exchange protocols against liveness properties constrained by Murray's refinement-closed interpretation of fairness provides a practical setting exemplifying the ability of our approach to verify properties that cannot, in general, be tested for via simple refinement checks in FDR. Typically, a Dolev-Yao (DY) intruder [6], which is limited by perfect cryptography but has complete control over the network, is assumed when analysing security protocols. However, such an intruder model trivially breaks liveness properties, as the DY intruder may choose not to communicate any message sent. Fair exchange protocols often rely upon the assumption that at least some messages are communicated using resilient channels [7], which eventually deliver each message [8]. In this paper we construct a CSP model of an intruder constrained by a resilient communication channel assumption, based on work by Cederquist and Dashti [9], that can be used to verify liveness properties in fair exchange protocols using the fairness constraints proposed by Murray [4].

Following the necessary background on CSP provided in Section 2, Section 3 describes how the atomic formulae of Lowe's temporal logic can be expressed using the temporal operators offered when LTL model checking in ProB. Section 3 can stand alone demonstrating how, in general, one can reason about liveness properties constrained by Murray's refinement-closed interpretation of fairness using ProB, while Section 4 describes its application in the specific setting of fair exchange protocols. In Section 4 we construct an intruder model for reasoning about liveness properties of fair exchange protocols in the presence of resilient channels. Finally, Section 5 and Section 6 describe related and future work.

2 CSP

CSP is a process algebra for describing models of interacting processes in terms of the atomic events that they perform [1]. Processes, denoted by identifiers beginning in uppercase (e.g., P, Q), interact by synchronising on visible events, denoted by lowercase characters (e.g., a, b). The set of all visible (i.e., external events) is denoted by Σ, which does not contain the internal action τ. *Stop* is the deadlocked process that performs no event. The CSP process $a \rightarrow P$ performs the event a and then acts as P. The equation $P = a \rightarrow P$ defines a recursive process that infinitely performs a. The process $P \,\Box\, Q$ may act as either P or Q, the choice of which is resolved by the environment. Similarly, the process $P \sqcap Q$

may act as either P or Q, but in this case the choice is resolved internally by the system. The difference between the internal and external choice operators is illustrated by processes P_3 and P_4 in Figure 1. The process $P \triangleright Q$ acts as P, although a timeout may occur, represented as an internal action τ, before P performs its first visible event. Following a timeout the process shall then act as Q, as illustrated by process P_5 in Figure 1.

The process $P \underset{A}{\parallel} Q$ runs P and Q in parallel, synchronising each occurrence of an event in A. Parallel composition $P \,|||\, Q$ of processes that do not synchronise on any event are said to be interleaved. Note that prefixing binds tighter than each of the choice operators, which in turn bind tighter than the parallel operators. The process $P \setminus A$ acts as P but with each of the events in A replaced by the internal action τ. Finally, $P[\![{}^{a}/_{b}]\!]$ acts as P but with each occurrence of event b in P replaced by event a.

Various semantic models of CSP [10] enable us to distinguish between processes. The coarsest model is the *traces* model $\mathcal{T}$, which captures the traces of events which a CSP process might exhibit. A sequence of visible events, $\langle e_1, e_2, \ldots, e_n \rangle$, is a *trace* of a process P if there is some execution of P in which exactly that sequence of events is performed. For example, the set of all traces of $P_1 = a \rightarrow \mathit{Stop} \,\Box\, b \rightarrow \mathit{Stop}$, which offers the environment a choice between performing a or b before reaching deadlock, is the set $\{\langle\rangle, \langle a \rangle, \langle b \rangle\}$. The same set of traces can be generated by the process $P_2 = a \rightarrow \mathit{Stop} \triangleright b \rightarrow \mathit{Stop}$, although in P_2 a choice between performing an a or b is not offered to the environment. Instead a may be performed unless a timeout first occurs, after which b may be performed. If an internal action τ may be performed from some state, then the state is *unstable* (e.g., the initial state of P_2) otherwise it is *stable*. A process may stabilise by performing successive internal actions until a stable state is reached. A process is divergent if it can perform an infinite succession of internal actions. In this paper we only consider systems that are free of divergence.

Lowe proved that the refusal-traces model $\mathcal{RT}$ is necessary for capturing requirements expressed in the temporal logic defined in [3]. For this reason $\mathcal{RT}$ is the semantic model used in the remainder of the paper. Rather than recording only the traces of events performed by a process, $\mathcal{RT}$ also records the set of actions refused after each event performed, $\langle X_0, e_1, X_1, e_1, X_2, \ldots, e_n, X_n \rangle$. As refusal sets are recorded only in stable states, the null refusal symbol, $\bullet$, is used to denote the absence of refusal information. A null refusal may be recorded should an event occur from an unstable state or should no attempt be made to observe the refusal information. For example, the CSP process P_2 has the refusal trace $\langle \bullet, a, \Sigma \rangle$. Likewise, $\langle \bullet, a, \Sigma \rangle$ is a refusal trace of the process P_1, although $\langle \emptyset, a, \Sigma \rangle$ is a refusal trace of P_1 but not of P_2.

In addition to these denotational semantics of CSP, there exists an operational semantics based upon labelled transition systems (LTS) [10]. Any CSP process can be given as an LTS, consisting of a non-empty set of states, an initial state, a set of labels $\Sigma \cup \{\tau\}$, and a set of labelled transitions, where a labelled transition $S \xrightarrow{a} S'$ denotes that an action a can be taken from the state S to move to state S'. All figures in this paper illustrate a CSP process as an LTS.

CSP has a theory of refinement that enables us to compare the behaviour of processes. If a process S is refined by a process P, then all of the possible behaviours of P must also be possible behaviours of S according to some semantic model, e.g., $S \sqsubseteq_{\mathcal{RT}} P$ states that P refines S in $\mathcal{RT}$. The refinement checker FDR [2] automatically checks whether a specification of a property (S) is satisfied by a proposed model (P). If the result of a check is negative, a refusal trace that leads to the violation of the property is given.

3 Model Checking under Fairness Constraints in ProB

As an alternative to refinement-based model checking, one may directly test whether requirements expressed in a temporal logic are satisfied by the model of the implementation. In [3], Lowe investigated the extent to which it can be checked that CSP processes satisfy temporal logic specifications using a refinement-based model checker, such as FDR. The following grammar, as proposed by Lowe, defines a temporal logic for specifying communicating processes:

$$\begin{aligned}\phi, \psi ::= {} & \mathit{true} \mid \mathit{false} \mid a \mid \mathit{available}\ a \mid \mathit{deadlocked} \mid \\ & \phi \wedge \psi \mid \phi \vee \psi \mid \neg \phi \mid \phi \Rightarrow \psi \mid \Diamond \phi \mid \Box \phi \mid \bigcirc \phi \mid \psi \,\mathcal{U}\, \phi \mid \phi \,\mathcal{R}\, \psi \qquad \text{where } a \in \Sigma\end{aligned}$$

Formulae in such a temporal logic regard properties of individual maximal paths of a process, i.e., an infinite refusals-trace of the process or a finite refusals-trace that ends in deadlock. $P \models \phi$ shall denote that a process P satisfies a formula ϕ if every maximal path of P satisfies ϕ. The formula a, for $a \in \Sigma$, states that the event a is guaranteed to be the first visible event performed. The formula *available* a states that the event a is not refused whenever the process stabilises before performing its first visible event, while *deadlocked* guarantees there is no next visible event. The logical operators $\wedge$, $\vee$ and $\Rightarrow$ have their usual meaning. The formulae $\Diamond \phi$ and $\Box \phi$ denote that eventually ϕ holds and that globally ϕ holds, respectively. $\bigcirc \phi$ guarantees that if there is a next visible action, ϕ holds after its occurrence. $\phi \,\mathcal{U}\, \psi$ states that ϕ remains true until ψ becomes true, whereas $\psi \,\mathcal{R}\, \phi$ states that ϕ remains true up to and including the state in which ψ becomes true, although ψ may never become true. The meaning each of these temporal operators is described more precisely in [3].

Lowe has shown that the temporal operators eventually $\Diamond$, until $\mathcal{U}$, and negation $\neg$ cannot in general be tested for via simple refinement checks. Furthermore, Murray has investigated the limits of refinement testing for CSP, demonstrating that there exist useful refinement-closed predicates that cannot in general be expressed as refinement checks in any standard CSP model that FDR can handle. Liveness properties under the assumption of strong or weak event fairness constitute one such class of predicates. It is common to make assumptions regarding which infinite behaviours of a system should be deemed fair when analysing an abstract model of a system. Many interpretations of fairness exist in the literature [11]; we shall follow Murray's interpretation of strong (resp. weak) event fairness assumption, which distinguishes itself from other notions of fairness [11–13] in its definition of *available*: an infinitely (resp. constantly) often *available* event shall occur infinitely often.

$$SEF = \bigwedge_{a \in \Sigma} (\Box \Diamond \mathit{available}\ a \Rightarrow \Box \Diamond a)$$
$$WEF = \bigwedge_{a \in \Sigma} (\Diamond \Box \mathit{available}\ a \Rightarrow \Box \Diamond a)$$

Alternative notions of fairness [11–13], defined in terms of 'available' (or enabled) events without checking the process stabilises before performing its first visible event, consider 'available' a to be satisfied by $a \rightarrow \mathit{Stop} \triangleright \mathit{Stop}$. Such a process is refined by *Stop* in each of the semantic models discussed in Section 2, in which a is clearly 'unavailable'. The advantage of Murray's interpretation of *SEF* and *WEF* is that they are refinement-closed in $\mathcal{RT}$. As temporal properties constrained by *SEF* or *WEF* cannot in general be expressed as refinement checks for FDR [4], we propose the use of ProB [14], which enables LTL model checking of CSP processes using the following grammar [5].

$$\phi, \psi ::= \texttt{true} \mid \texttt{false} \mid \texttt{[a]} \mid \texttt{e(a)} \mid \texttt{deadlock} \mid \phi\ \texttt{\&}\ \psi \mid \phi\ \texttt{or}\ \psi \mid \texttt{not}\ \phi \mid \phi\ \texttt{=>}\ \psi \mid \texttt{F}\,\phi \mid \texttt{G}\,\phi \mid \texttt{X}\,\phi \mid \phi\,\texttt{U}\,\psi \mid \psi\,\texttt{R}\,\phi \quad \text{where } \texttt{a} \in \Sigma \cup \{\tau\}$$

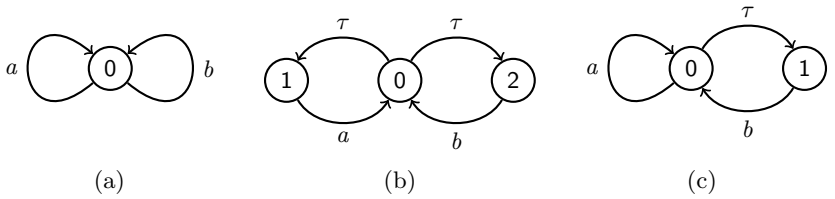
Not all of the above operators offered by ProB for LTL model checking match their counterpart in the grammar defined by Lowe. The temporal operators $\Diamond$, $\Box$, $\mathcal{U}$ and $\mathcal{R}$ can be expressed directly as their counterpart in ProB, i.e., `F`, `G`, `U` and `R`. The same is true for the boolean logic operators. However, Lowe's operators a, *available* a, *deadlocked* and $\bigcirc \phi$ require further attention.

Note that Lowe's actions range over the set of visible actions, whereas ProB also enables us to express properties in terms of internal events. The formula `[a]` is satisfied along some linear execution path of a process if the first action taken is an a. This is true also of the internal action τ, i.e., `[tau]` is satisfied along some path if the first action taken from the current state along the path is the invisible action. A path satisfies `e(a)`, where $a \in \Sigma \cup \{\tau\}$, if a is not refused from the current state, whether a is the next action taken along the path or not. $\texttt{X}\,\phi$ states that there is a visible or internal action leading to a next state, in which ϕ holds. Finally, `deadlock` is satisfied by a state from which no visible or internal actions are offered.

Despite their differences, we can express the four atomic formulae from Lowe's temporal logic that do not match their counterpart in the grammar offered by ProB in the following manner:

a	$\rightarrow$ `[tau] U [a]`
available a	$\rightarrow$ `[tau] U ((e(tau) & not [tau]) or (e(a) & not e(tau)))`
deadlocked	$\rightarrow$ `[tau] U deadlock`
$\bigcirc\phi$	$\rightarrow$ `[tau] U (deadlock or (not [tau] & X` ϕ`))`

We captured the formula a, that states the event a is guaranteed to be the first visible event performed, by asserting that τ events are performed from each state on the path until an a is performed. Similarly, *deadlocked*, which guarantees there is no next visible event, is expressed by asserting that τ events are performed from each state along the path until `deadlock` is reached. The availability of a, defined as a is enabled whenever the process stabilises before



(a) (b) (c)

Fig. 1: CSP Processes: (a) $P_3 = a \rightarrow P_3 \,\Box\, b \rightarrow P_3$; (b) $P_4 = a \rightarrow P_4 \sqcap b \rightarrow P_4$; and (c) $P_5 = a \rightarrow P_5 \triangleright b \rightarrow P_5$

performing its first event, is captured by stating that τ events are performed, until either some visible action is taken from an unstable state or, if, a stable state is reached, then a must be enabled. Finally, that ϕ is guaranteed to hold after the first visible action is captured as `[tau] U (deadlock or (not [tau] & X`ϕ`))`. Note that $\bigcirc\phi$ is vacuously true if there is no next stable state. The stronger statement 'there exists a next state and in it ϕ holds' is expressed by removing `deadlock` from the formula.

To illustrate his interpretation of strong event fairness, and prove that it cannot be expressed as a simple refinement check for FDR, Murray had us consider the three CSP processes depicted in Figure 1. As a is globally available in P_3, infinite executions in which an a never occurs are considered *unfair*. Likewise, infinite executions of P_3 in which a b never occurs are also unfair. Hence, $P_3 \models \mathit{SEF} \Rightarrow \Diamond a$ and $P_3 \models \mathit{SEF} \Rightarrow \Diamond b$.

Neither *available* a nor *available* b is satisfied in the initial state of P_4, which is visited infinitely many times in each infinite path. Thus, neither a nor b is globally available and so no infinite path of P_4 is considered unfair. Hence, $P_4 \not\models \mathit{SEF} \Rightarrow \Diamond a$ and $P_4 \not\models \mathit{SEF} \Rightarrow \Diamond b$. In process P_5, a can only occur from an unstable state, so infinite paths in which a never occurs must be deemed *fair*. In each state of P_5, whenever the process stabilises before performing its first visible event, only b is offered. Therefore, infinite paths of P_5 in which a never occurs must be deemed *fair*, whereas infinite paths in which b never occurs must be deemed *unfair*. Hence, $P_5 \not\models \mathit{SEF} \Rightarrow \Diamond a$, but $P_5 \models \mathit{SEF} \Rightarrow \Diamond b$.

Processes P_3, P_4 and P_5 can be checked against $\mathit{SEF} \Rightarrow \Diamond a$ and $\mathit{SEF} \Rightarrow \Diamond b$ in ProB using our definitions of a and *available* a above. The same results hold under Murray's interpretation of weak event fairness [4], which was also proved to not be expressible as a simple refinement check for FDR. Thus our definitions of a and *available* a also enable one to use ProB to check CSP processes against properties constrained by weak event fairness.

4 An Intruder Model in CSP for Verifying Liveness

When analysing security protocols it is standard to assume a Dolev-Yao (DY) intruder model in which the intruder has full control over the network [6]. Liveness properties are not satisfiable under this assumption, as the DY intruder

may choose not to communicate any message sent. Fair exchange protocols rely upon the assumption that at least some messages are communicated using resilient channels in order to satisfy certain liveness properties [7]. In this section we construct a new intruder model for reasoning about liveness properties of fair exchange protocols in the presence of resilient channels. We analyse our models against fairness constrained properties in ProB, as described in Section 3.

Our intruder model is based upon the one of Cederquist and Dashti [9], who analysed μCRL [15] models of fair exchange protocols against μ-calculus expressions [16] of liveness properties constrained by fairness using CADP [17]. While the properties against which Cederquist and Dashti analysed their models are not refinement-closed, our liveness properties shall be. Should our models satisfy some liveness property constrained under Murray's *SEF*, then any refinement of our models in $\mathcal{RT}$ shall also necessarily satisfy the property.

We begin, in Section 4.1, with a description of a CSP model of the DY intruder proposed for reasoning about safety properties. We shall demonstrate why such an intruder model is insufficient for reasoning about liveness properties. In Section 4.2, we discuss how this issue was addressed by Cederquist and Dashti [9], but show that their properties are not refinement-closed. We address this issue in Section 4.3 by revising Roscoe's intruder model to enable the intruder to refuse to perform certain events. Finally, in Section 4.4, we propose a CSP model of the intruder that enables us to analyse fair exchange protocols in the presence of resilient channels with use of the fairness constraints described in Section 3.

4.1 Roscoe's Intruder Model for Verifying Safety

In [18], Roscoe's lazy spy was constructed to check whether the Needham-Schroeder public-key protocol satisfies certain safety properties in the presence of a DY intruder. Although written differently to better suit model checking in FDR, Roscoe's spy is $\mathcal{RT}$ equivalent to the following CSP process. The function $Close(X)$, defined in [18], returns the closure of a set of facts X under all deductions that the intruder can perform. The set $\mathcal{M}$ contains all the messages that can be sent or received in the protocol.

$$Spy^S(X) = \left(\begin{array}{c} \Box \quad say.m \rightarrow Spy^S(X) \\ m \in X \cap \mathcal{M} \\ \Box \quad \Box \quad learn.m \rightarrow Spy^S(Close(X \cup \{m\})) \\ m \in \mathcal{M} \end{array} \right)$$

It is important to appreciate that Spy^S is willing to accept any incoming message on *learn* and is always willing to *say* any message it can construct as a consequence of the use of external choice in its construction. Spy^S cannot refrain from saying a message it can construct, should some other process synchronise on performing such an event. This is of little concern when considering only safety properties, as checked in the traces model of CSP, in which the intruder has little to gain by refraining from performing certain events. However, the same is not true when checking liveness properties. Let us consider the following

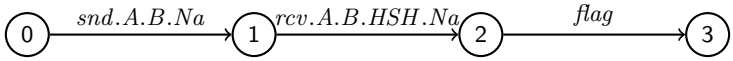


Fig. 2: SYS_1

system, similar to that described in [13], where $\mathcal{A} = \{A, B, I\}$ with A and B being the honest agents and I the intruder.

$$
\begin{aligned}
SND_1 &= snd.A.B.Na \rightarrow Stop \\
RCV_1 &= rcv.A.B.HSH.Na \rightarrow \mathit{flag} \rightarrow Stop \\
SYS_1 &= (SND_1 \,|||\, RCV_1) \underset{\{|snd,rcv|\}}{\parallel} Spy^S [\![{}^{snd.x.y,rcv.y.x}/_{learn,say} \,|_{x \leftarrow \mathcal{A}\setminus\{I\}, y \leftarrow \mathcal{A}\setminus\{x\}}]\!]
\end{aligned}
$$

Spy^S allows any message to be sent by the other agents, and is willing to deliver any message it can construct from its current knowledge. We need only assume appropriate deduction rules for hashing and sequence generation for our examples, and in each the initial knowledge of the intruder is assumed empty. Figure 2 illustrates SYS_1, in which SND_1 and RCV_1 synchronise with Spy^S on events in $\{|\, snd, rcv \,|\} = \{snd.a.b.m, rcv.a.b.m \mid a \leftarrow \mathcal{A}, b \leftarrow \mathcal{A}\setminus\{a\}, m \leftarrow \mathcal{M}\}$. The sender, A, who behaves as the process SND_1, is willing to send a single message consisting only of a nonce, Na, to B. Conversely, the process RCV_1, which expresses the behaviour of the recipient B, is willing only to receive the hashed value $HSH.Na$. Clearly, A and B shall fail to communicate should they attempt to do so over any reasonable medium, as B is willing to receive none of the messages sent by A. However, the parallel composition of SND_1 and RCV_1 with the intruder process, Spy^S, fabricates curious behaviour. The signal event, flag, occurs in all maximal paths of SYS_1 only because the modelling of the intruder guarantees that it shall.

4.2 Cederquist-Dashti Resilient Channel Assumption

In [13], Dashti demonstrated that the fairness constraint that states that 'each infinitely enabled transition is infinitely taken' is insufficient to resolve this issue. It is important to appreciate the subtle differences between this fairness assumption and Murray's SEF, as described in Section 3. Firstly, Dashti's initial fairness assumption is described in terms of transitions, whereas Murray's is described in terms of events. Secondly, external and internal events are treated the same in Dashti's initial fairness assumption, but not in Murray's. Hence, the process P_5, illustrated in Figure 1, satisfies $\Diamond a$ under Dashti's interpretation of fairness but not Murray's.

Regardless of their differences, under either of these fairness constraints, flag is guaranteed to eventually occur in SYS_1 only because the modelling of the intruder guarantees that it shall. To resolve this issue, Cederquist and Dashti [9] proposed an alternative fairness constraint and to parameterise the intruder process by the set of all messages sent but not yet delivered, which we shall denote as Y. Initially the set is empty, but following each snd action the message,

as well as the correct addressing information, is added to Y. Should a message be sent multiple times without being delivered, only one instance of the message is recorded within Y. This intruder satisfies Cederquist and Dashti's resilient communication channel assumption RCC_1 [9]. This assumption states that a resilient channel delivers the message at least once after the message has been sent multiple times.

As an alternative to Dashti's initial fairness assumption described above, Cederquist and Dashti propose the use of the μ-calculus notion of fair reachability of an action, which states that whatever path has been taken previously, there remains a path in which the action finally occurs. By distinguishing those messages sent but not yet received and all others messages delivered by the intruder as rcv and $rcv^\dagger$, respectively, Cederquist and Dashti check that whatever path has been taken previously, there remains a path containing no $rcv^\dagger$'s in which the signal event *flag* finally occurs. Such a property is not refinement-closed. Consider again the examples illustrated in Figure 1. In each process P_3, P_4 and P_5, no matter what path has been taken previously, there always exist a path in which a occurs. However, removing the nondeterminism from P_4, for example by demanding that the b path is always taken, refines P_4 in each semantic model discussed in Section 2. In particular, $P_4 \sqsubseteq_{\mathcal{RT}} Q$ where $Q = b \rightarrow Q$. Hence, although P_4 satisfies fair reachability, after refinement of P_4 this property may no longer be satisfied.

4.3 An Intruder Model Without Resilient Channels

To maintain the use of CSP's theory of refinement, we wish to analyse our models against liveness properties expressed in Lowe's temporal logic constrained by Murray's *SEF*, as presented in Section 3. Therefore, rather than adopting an alternative fairness constraint, we have adapted the model of the intruder to include behaviour enabling the intruder to refuse to cooperate. Our intruder model enables the intruder to nondeterministically choose the set of messages he is willing to send at any given time, from the set of all subsets of the messages he can construct at that time. CSP's notion of external choice and the use of the semantic model, $\mathcal{RT}$, are well-suited to modelling the intruder behaviour in this way. We begin by revising the process Spy^S to include behaviour enabling the intruder to refuse to cooperate in this manner.

$$Spy^{L_0}(X) = \bigsqcap_{\mathcal{S} \in Set(X \cap \mathcal{M})} \left(\begin{array}{c} \Box_{m \in \mathcal{S}} \; say.m \rightarrow Spy^{L_0}(X) \\ \Box \; \Box_{m \in \mathcal{M}} \; learn.m \rightarrow Spy^{L_0}(Close(X \cup \{m\})) \end{array} \right)$$

The process $Spy^{L_0}(X)$ presents an initial attempt at modelling the intruder's ability to refuse to cooperate. However, the result of the renaming within the process $Spy^{L_0}(X)[\![^{snd.x.y,rcv.y.x}/_{learn,say} \; |_{x \leftarrow \mathcal{A} \setminus \{I\}, y \leftarrow \mathcal{A} \setminus \{x\}} \;]\!]$ is such that the intruder chooses whether or not he is willing to perform a rcv event regarding (for example) Na. It does not give the intruder the ability to offer $rcv.A.B.Na$

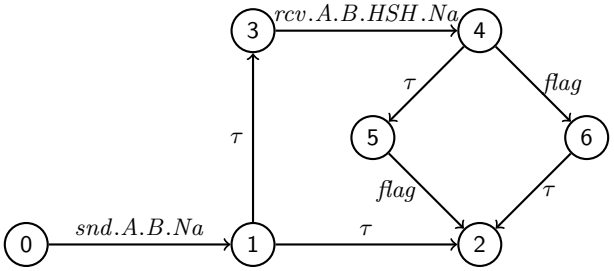


Fig. 3: $(SND_1 \;|||\; RCV_1) \underset{\{|snd,rcv|\}}{\parallel} Spy^L$

whilst also refusing $rcv.B.A.Na$. The process $Spy^L(X)$, constructed directly of snd and rcv events, provides this necessary additional behaviour.

$$Spy^L(X) = \prod_{\mathcal{S} \in Set\left(\left\{ (x,y,n) \middle| \substack{x \leftarrow \mathcal{A} \\ y \leftarrow \mathcal{A} \setminus \{x,I\} \\ n \leftarrow (X \cap \mathcal{M})} \right\}\right)} \left(\begin{array}{ll} \Box_{(a,b,m) \in \mathcal{S}} & rcv.a.b.m \rightarrow Spy^L(X) \\ \Box \; \Box_{(a,b,m) \in \left\{ (x,y,n) \middle| \substack{y \leftarrow \mathcal{A} \\ x \leftarrow \mathcal{A} \setminus \{y,I\} \\ n \leftarrow \mathcal{M}} \right\}} & snd.a.b.m \rightarrow \\ & Spy^L(Close(X \cup \{m\})) \end{array} \right)$$

Replacing Spy^S with Spy^L in SYS_1 produces the state space illustrated in Figure 3. From the initial state the intruder does not hold enough knowledge to generate any messages, so $snd.A.B.Na$ is guaranteed to be the first action performed. Following $snd.A.B.Na$, the intruder chooses which subset of messages he is willing to deliver from the set of all messages he can construct. Hence, he chooses between performing $rcv.A.B.HSH.Na$ or refusing to perform $rcv.A.B.HSH.Na$. If he chooses to refuse the event, then no further actions are possible, otherwise $rcv.A.B.HSH.Na$ is performed in synchrony with RCV_1. RCV_1 is then able to perform the event *flag*, while the intruder can again resolve his internal choice. Thus the interleaving of *flag* and τ actions means that they may occur in either order before the system deadlocks. As a consequence of our revisions to the intruder model, the signal event *flag* no longer occurs in all maximal paths of SYS_1, as was desired.

With full control over the network, the DY intruder may now choose not to communicate any message sent. As a result, liveness properties, such as $\Diamond$*flag*, can never be guaranteed in this model, even under the assumption of strong event fairness. It is for this reason that fair exchange protocols rely upon the assumption that at least some messages are communicated over resilient channels [7]. Our revised intruder, which can choose not to communicate any message, requires further revision to capture Cederquist and Dashti's resilient channel assumption, RCC_1. First let us motivate such a revision to the intruder model via a second example.

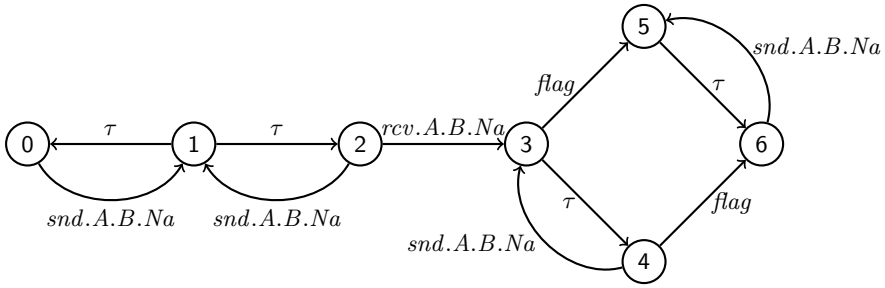


Fig. 4: SYS_2

$$
\begin{aligned}
SND_2 &= snd.A.B.Na \rightarrow SND_2 \\
RCV_2 &= rcv.A.B.Na \rightarrow flag \rightarrow Stop \\
SYS_2 &= (SND_2 \,|||\, RCV_2) \underset{\{|snd,rcv|\}}{\parallel} Spy^L
\end{aligned}
$$

Figure 4 illustrates the state space of such a system. Initially, $snd.A.B.Na$ is guaranteed to be the first action performed, as the intruder does not hold enough knowledge to generate any messages. Following $snd.A.B.Na$, the intruder again chooses which messages he is willing to deliver from the set of all subsets of the messages he can construct. Hence, he chooses between performing or refusing to perform $rcv.A.B.Na$. If he chooses to refuse the event, then the system returns to the initial state, otherwise either $snd.A.B.Na$ is again performed in synchrony with SND_2, returning to a previously visited state, or $rcv.A.B.Na$ is performed in synchrony with RCV_2. In the latter case, RCV_2 is able to perform the event *flag*, while the intruder can again resolve his internal choice. Thus, the interleaving of the *flag* and τ actions means that they may occur in either order, after which $snd.A.B.Na$ is the only possible event, in each case returning the system to a previously visited state.

Using ProB we can verify that SYS_2 does not satisfy $\Diamond flag$. The system includes infinite execution paths in which (i) the intruder always chooses to refuse the delivery of Na, and (ii) the intruder is willing to perform $rcv.A.B.Na$, but infinitely often $snd.A.B.Na$ is taken instead.

4.4 An Intruder Model With Resilient Channels

We shall revise the intruder model such that he is unable to refuse to deliver messages that have been sent but not yet delivered. Such a revised model can then be analysed under the assumption of strong event fairness.

$$
\begin{aligned}
&Spy^{L\dagger}(X, Y) = \\
&\prod_{\mathcal{S} \in Set\left(\left\{ (x,y,n) \middle| \substack{x \leftarrow \mathcal{A} \\ y \leftarrow \mathcal{A} \setminus \{x,I\} \\ n \leftarrow (X \cap \mathcal{M}) \setminus Y} \right\}\right)} \left(\begin{array}{l} \Box_{(a,b,m) \in \mathcal{S} \cup Y} \; rcv.a.b.m \rightarrow Spy^L(X, Y \setminus \{(a,b,m)\}) \\ \Box \; \Box_{(a,b,m) \in \left\{ (x,y,n) \middle| \substack{y \leftarrow \mathcal{A} \\ x \leftarrow \mathcal{A} \setminus \{y,I\} \\ n \leftarrow \mathcal{M}} \right\}} \; \begin{array}{l} snd.a.b.m \rightarrow \\ Spy^L(Close(X \cup \{m\}), Y \cup \{(a,b,m)\}) \end{array} \end{array} \right)
\end{aligned}
$$

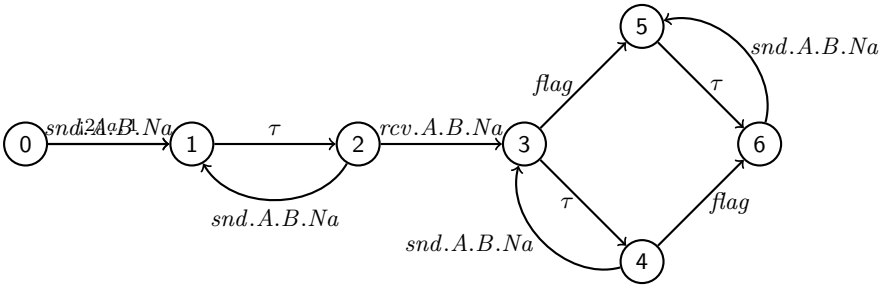


Fig. 5: $(SND_2 \;|||\; RCV_2) \underset{\{|snd,rcv|\}}{\parallel} Spy^{L\dagger}$

Like Cederquist and Dashti's intruder model, our process $Spy^{L\dagger}$ is parameterised by the set Y of all messages sent but not yet delivered. Initially the set is empty. Following each $snd.a.b.m$ event, the message as well as the correct addressing information (a, b, m) is added to Y. Should a message be sent multiple times without being delivered, only one instance of the message is recorded within Y, as this is the information required to assure the intruder satisfies RCC_1. As the correct addressing information must be recorded in Y, $Spy^{L\dagger}(X, Y)$ is constructed explicitly of *snd* and *rcv* events, rather than renaming a process constructed of simpler *learn* events. The consequence of the other revisions is that the intruder never refuses the delivery of messages contained in Y, but remains able to refuse any subset of the remaining messages he can construct. We have used ProB to successfully check that the revised model SYS_2, as illustrated in Figure 5, satisfies $SEF \Rightarrow \Diamond\, flag$ using the definitions from Section 3.

4.5 Conclusion

The intruder model $Spy^{L\dagger}$ constructed in this section is suitable for analysing fair exchange protocols in the presence of resilient channels and a DY intruder. Our analyses needed to be closed under refinement to be sure that the system will be secure under any refinements of the nondeterministic intruder, which represent different attack strategies. Murray's refinement-closed interpretation of strong event fairness *SEF* was added as a premise of the liveness property being checked. The consequence of this fairness constraint was that paths in which the intruder never delivers some message from Y, supposing that the recipient is always willing to accept it, were disregarded as unfair paths.

A fair exchange protocol may only require certain messages to be sent via resilient channels. Minimal changes to Spy^{L} are required to model such a system. Distinguishing between messages sent over non-resilient and resilient channels as *snd* and *sndr*, respectively, is sufficient to record which messages should be contained in Y.

5 Related Work

Building upon Schneider's analysis [19] and drawing from the work of Evans [20], Wei and Heather analysed the Zhou-Gollmann non-repudiation protocol [21] protocol using CSP and FDR [22] as well as the theorem prover PVS [23]. The contribution that distinguishes our CSP intruder model from those that preceded it is the modelling of the resilient channel. Wei and Heather modelled the resilient communication between the trusted third party and the other agents as a synchronous communication over a reliable and secure channel. Originally, Zhou and Gollmann assumed that the channel, over which the third party sends messages, will eventually be available, i.e., that there exists a resilient channel between the third party and other agents. Our intruder model increases the possible behaviours of the intruder, who is able to record which messages are sent over the resilient channel and delay the delivery of such messages. He must only not indefinitely delay the delivery of such messages, assuming the recipient remains willing to receive it. Thus our intruder model aims to more faithfully capture the assumptions of resilience in CSP models of fair exchange protocols.

By adopting Murray's refinement-closed interpretation of fairness [4] we maintain CSP's theory of refinement. Furthermore, such a notion of fairness is weaker than the one of Cederquist and Dashti [9]. Rather than treating all outgoing external and internal actions fairly, we demand only that infinitely often available (visible) events are treated fairly. As we adopt a weaker notion of fairness, our intruder model is stronger, because less of its behaviour is restricted. Even under this stronger intruder model we have been successful in checking the satisfaction of liveness properties in the examples given in this paper.

The Process Analysis Toolkit [24] has specific functionality for model checking CSP processes under various fairness assumptions. However, such notions of fairness do not match the ones proposed by Murray and are not refinement-closed. The process P_4, illustrated in Figure 1, satisfies $\Diamond b$ under the PAT interpretation of strong event fairness, but it also satisfies $\Diamond a$ as the internal event τ is treated the same as visible actions. Refining P_4 by removing the unstable a action would cause the process to no longer satisfy $\Diamond a$, so PAT's interpretation of strong event fairness is not refinement-closed.

6 Discussion and Future Work

In [4] it was shown that Murray's refinement-closed interpretation of strong and weak event fairness cannot be expressed as refinement checks in the semantic models supported by FDR. We therefore propose the use of ProB, as we have shown that its LTL model checker [5] can be used to model check liveness properties under such fairness constraints, which were expressed in Lowe's temporal logic [3]. More generally, we have shown how any formula of Lowe's temporal logic can be expressed in the grammar offered by ProB for LTL model checking, even though the two grammars differ.

Subsequently, using Roscoe's intruder model [18] as a foundation, we proposed a new intruder model for reasoning about liveness properties in security

protocols in the presence of resilient channels. Roscoe's intruder model [18] required revision for this purpose, since this model of the intruder can drop all messages, thus trivially violating all liveness properties. Furthermore, an example was provided to demonstrate that Roscoe's intruder model can in some sense help in satisfying liveness properties that would not otherwise be satisfied in the presence of a reliable medium. Such observations were first made in [13], which justified the construction of a new intruder model in [9]. The primary distinction between our approach to that in [9] is the use of Murray's refinement-closed interpretation of strong event fairness. Our analyses needed to be closed under refinement to be sure that the system will be secure under any and all refinements of the nondeterministic intruder, which represent different attack strategies.

The examples provided in this paper to demonstrate and justify our approach were necessarily simplistic to enable ProB to complete the checks in reasonable time. The application of our approach to more meaningful fair exchange protocols, such as those described in [7], remains future research. Murray's fairness constraint was added as a premise of the liveness property, i.e., $SEF \Rightarrow \phi$, where SEF was constructed as the conjunction over the potentially large set of events Σ. This method of model checking under fairness constraints is inefficient, as the time complexity of LTL model checking is exponential in the size of the formula [24]. Rather than incorporating the fairness constraint as a premise of the property being checked, dedicated algorithms have been implemented within PAT to analyse CSP models against LTL properties under fairness. In Section 5 we demonstrated that PAT's interpretation of strong event fairness is not refinement-closed, so investigating how to efficiently check liveness properties under SEF remains an open research question.

When model checking CTL, fairness constraints cannot simply be added as a premise of the property being checked, as they are not typically expressible in branching time logic. It is worth considering how the work in [3] can be adapted to check CSP processes against CTL properties that are closed under refinement. Furthermore, it would be worthwhile to consider the use of more expressive temporal logics when model checking CSP processes. The relationship between LTL, CTL, CTL* and μ-calculus is well documented, but it is unclear precisely how the limits of refinement testing for model checking CSP processes relate.

Acknowledgements This work is supported by the Netherlands Organisation for Scientific Research (NWO). Our thanks go to Mohammad Torabi Dashti and Toby Murray for discussions relating to their work as well as comments provided on previous drafts of this paper. We also thank Gavin Lowe for discussing his paper. We would like to thank the ProB team, in particular Ivaylo Dobrikov, for providing notable tool support. Finally, we would like to thank the anonymous reviewers for their constructive comments.

References

1. Hoare, C.A.R.: Communicating Sequential Processes. Prentice-Hall (1985)
2. Gardiner, P., Goldsmith, M., Hulance, J., Jackson, D., Roscoe, B., Scattergood, B., Armstrong, P.: FDR Manual. Oxford University. (2010)
3. Lowe, G.: Specification of communicating processes: Temporal logic versus refusals-based refinement. Formal Aspects of Computing **20** (2008) 277–294
4. Murray, T.: On the limits of refinement-testing for model-checking CSP. Formal Aspects of Computing (2012) To appear.
5. Plagge, D., Leuschel, M.: Seven at one stroke: LTL model checking for high-level specifications in B, Z, CSP, and more. Software Tools for Technology Transfer **12** (2010) 9–21
6. Dolev, D., Yao, A.: On the security of public key protocols. IEEE Transactions on Information Theory **29** (1983) 198–208
7. Kremer, S., Markowitch, O., Zhou, J.: An intensive survey of fair non-repudiation protocols. Computer Communications **25** (2002) 1606–1621
8. Asokan, N.: Fairness in electronic commerce. Technical report, University of Waterloo (1998)
9. Cederquist, J., Torabi Dashti, M.: An intruder model for verifying liveness in security protocols. In: Proc. FMSE'06, ACM (2006) 23–32
10. Roscoe, A.: Understanding Concurrent Systems. Springer (2010)
11. Francez, N.: Fairness. Springer (1986)
12. Puhakka, A., Valmari, A.: Liveness and fairness in process-algebraic verification. CONCUR 2001 (2001) 202–217
13. Torabi Dashti, M.: Keeping Fairness Alive. PhD thesis, VU University Amsterdam (2008)
14. Leuschel, M., Butler, M.: ProB: An automated analysis toolset for the B method. Software Tools for Technology Transfer **10** (2008) 185–203
15. Fokkink, W.J.: Modelling Distributed Systems. Springer (2007)
16. Mateescu, R., Sighireanu, M.: Efficient on-the-fly model checking for regular alternation-free mu-calculus. Science of Computer Programming **46** (2003) 255–281
17. Garavel, H., Lang, F., Mateescu, R., Serwe, W.: CADP 2010: A toolbox for the construction and analysis of distributed processes. In: Proc. TACAS'11. Volume 6605 of LNCS., Springer (2011) 372–387
18. Roscoe, A.W.: The Theory and Practice of Concurrency. Prentice-Hall (1998)
19. Schneider, S.: Formal analysis of a non-repudiation protocol. In: Proc. CSF'98, IEEE (1998) 54–65
20. Evans, N., Schneider, S.: Verifying security protocols with PVS: Widening the rank function approach. Journal of Logic and Algebraic Programming **64** (2005) 253–284
21. Zhou, J., Gollman, D.: A fair non-repudiation protocol. In: S&P'96, IEEE (1996) 55–61
22. Wei, K., Heather, J.: Towards verification of timed non-repudiation protocols. In: Proc. FAST'05. Volume 3866 of LNCS., Springer (2006) 244–257
23. Wei, K., Heather, J.: A theorem-proving approach to verification of fair non-repudiation protocols. In: Proc. FAST'06. Volume 4691 of LNCS., Springer (2007) 202–219
24. Sun, J., Liu, Y., Dong, J.S., Pang, J.: PAT: Towards flexible verification under fairness. In: Proc. CAV'09. Volume 5643 of LNCS., Springer (2009) 709–714