Reducing complex CSP models to traces via priority

David Mestel[1] A.W. Roscoe[2]

Department of Computer Science, University of Oxford

Abstract

Hoare's Communicating Sequential Processes (CSP) [6] admits a rich universe of semantic models. In this paper we study finite observational models, of which at least six have been identified for CSP, namely traces, failures, revivals, acceptances, refusal testing and finite linear observations [11]. We show how to use the recently-introduced *priority* operator ([12], ch.20) to transform refinement questions in these models into trace refinement (language inclusion) tests. Furthermore, we are able to generalise this to any (rational) finite observational model. As well as being of theoretical interest, this is of practical significance since the state-of-the-art refinement checking tool FDR3 [4] currently only supports two such models.

Keywords: CSP, denotational semantics, priority

1 Introduction

A number of different forms of process calculus have been developed for the modeling of concurrent programs, including Hoare's Communicating Sequential Processes (CSP) [6], Milner's Calculus of Communicating Systems (CCS) [7], and the π-calculus [8]. Unlike the latter two, CSP's semantics are traditionally given in behavioural semantic models coarser than bisimulation.

In this paper, we study finite linear-time observational models for CSP; that is, models where all observations considered can be determined in a finite time by an experimenter who can see the visible events a process communicates and the sets of events it can offer in any stable state. While the experimenter can run the process arbitrarily often, he or she can only record the results of individual finite executions. Thus each behaviour recorded can be deduced from a single finite sequence of events together with the sets of events accepted in stable states during and immediately after this *trace*.

At least six such models have been considered for CSP, but the state-of-the art refinement checking tool, FDR3 [4], currently only supports two, namely *traces*

[1] Email: david.mestel@cs.ox.ac.uk

[2] Email: bill.roscoe@cs.ox.ac.uk

This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

and *failures* (it also supports the failures-divergences model, which is not finite observational).

We present a construction which produces a context $\mathcal{C}$ such that refinement questions in the failures model correspond to trace refinement questions under the application of $\mathcal{C}$. We are able to generalise this to show (Theorem 5.4) that a similar construction is possible not only for the six models which have been studied, but also for any sensible finite observational model (where 'sensible' means that the model can be recognised by a finite-memory computer, in a sense which we shall make precise).

We first briefly describe the language of CSP. We next give an informal description of our construction for the failures model. To prove the result in full generality, we first give a formal definition of a finite observational model, and of the notion of rationality. We then describe our general construction. Finally we discuss performance and optimisation issues.

2 The CSP language

We provide a brief outline of the language, largely taken from [11]; the reader is encouraged to consult [12] for a more comprehensive treatment.

Throughout, Σ is taken to be a finite nonempty set of communications that are visible and can only happen when the observing environment permits via handshaken communication. The actions of every process are taken from $\Sigma \cup \{\tau\}$, where τ is the invisible internal action that cannot be prevented by the environment. Note that the usual treatment of CSP permits sequential composition by including another un-preventable event ✓ to represent termination; this adds slight complications to each model and we omit it for simplicity. It could be added back without any significant alteration to the results of this paper.

The constant processes of CSP are

- *STOP* which does nothing—a representation of deadlock.
- **div** which performs (only) an infinite sequence of internal τ actions—a representation of divergence or livelock.
- *CHAOS* which can do anything except diverge.

The prefixing operator introduces communication:

- $a \to P$ communicates the event a before behaving like P.

There are two forms of binary choice between a pair of processes:

- $P \sqcap Q$ lets the process decide to behave like P or like Q: this is *nondeterministic* or *internal* choice.
- $P \,\Box\, Q$ offers the environment the choice between the initial Σ-events of P and Q. If the one selected is unambiguous then it continues to behave like the one chosen; if it is an initial event of both then the subsequent behaviour is nondeterministic. The occurence of τ in one of P and Q does *not* resolve the choice (unlike CCS +). This is *external* choice.

We only have a single parallel operator in our core language since all the usual ones of CSP can be defined in terms of it as discussed in Chapter 2 etc. of [12].

- $P \underset{X}{\|} Q$ runs P and Q in parallel, allowing each of them to perform any action in $\Sigma \setminus X$ independently, whereas actions in X must be synchronised between the two.

There are two operators that change the nature of a process's communications.

- $P \setminus X$, for $X \subseteq \Sigma$, *hides* X by turning all P's X-actions into τs.
- $P[\![R]\!]$ applies the *renaming* relation $R \subseteq \Sigma \times \Sigma$ to P: if $(a, b) \in R$ and P can perform a, then $P[\![R]\!]$ can perform b. The domain of R must include all visible events used by P. Renaming by the relation $\{(a, b)\}$ is denoted $[\![^a/_b]\!]$.

There is another operator that allows one process to follow another:

- $P \Theta_A Q$ behaves like P until an event in the set A occurs, at which point P is shut down and Q is started. This is the *throw* operator.

The final CSP construct is *recursion*: this can be single or mutual (including mutual recursions over infinite parameter spaces), can be defined by systems of equations or (in the case of single recursion) in line via the notation $\mu\, p.P$, for a term P that may include the free process identifier p. Recursion can be interpreted operationally as having a τ-action corresponding to a single unwinding. Denationally, we regard P as a function on the space of denotations, and interpret $\mu\, p.P$ as the least fixed point of this function.

We also make use of the *interleaving* operator $|||$, which allows processes to perform actions independently and is equivalent to $\underset{\emptyset}{\|}$, and the process RUN_X, which always offers every element of the set X and is defined by $RUN_X = \Box_{x \in X}\, x \to RUN_X$.

2.1 Priority

The prioritisation operator is discussed in detail in Chapter 20 of [12]. It allows us to specify an ordering on the set of visible events Σ, and prevents lower-priority events from occuring whenever a higher-priority event or τ is available.

The operator described in [12] as implemented in FDR3 [4] is parametrised by three arguments: a process P, a partial order $\leq$ on the event set Σ, and a subset $X \subseteq \Sigma$ of events that can occur when a τ is available. We require that all elements of X are maximal with respect to $\leq$. Writing $\mathit{initials}(P) \subseteq \Sigma \cup \{\tau\}$ for the set of events that P can immediately perform, and extending $\leq$ to a partial order on $\Sigma \cup \{\tau\}$ by adding $y \leq \tau\, \forall\, y \in \Sigma \setminus X$, we define the operational semantics of prioritise as follows:

$$\frac{P \xrightarrow{a} P' \wedge \forall\, b \neq a. a \leq b \Rightarrow b \notin \mathit{initials}(P)}{\text{prioritise}(P, \leq, X) \xrightarrow{a} \text{prioritise}(P', \leq, X)}\ (a \in \Sigma \cup \{\tau\}).$$

Note that prioritise is not compositional over denotational models other than the most precise model $\mathcal{FL}$, so we think of it as an optional addition to CSP rather than an integral part of it; when we refer below to particular types of observation as giving rise to valid models for CSP, we will mean CSP without priority.

3 Example: the failures model

We first demonstrate our construction using the *failures* model: we will produce a context $\mathcal{C}$ such that for any processes P, Q, we have that Q refines P in the failures model if and only $\mathcal{C}[Q]$ refines $\mathcal{C}[P]$ in the traces model.

3.1 The traces and failures models

The *traces* model $\mathcal{T}$ is familiar from automata theory, and represents a process by the set of (finite) strings of events it is able to accept. Thus each process is associated (for fixed alphabet Σ) to a subset of Σ^* the set of finite words over Σ.

The *failures* model $\mathcal{F}$ also records sets X of events that the process is able to stably refuse after a trace s (that is, the process is able after trace s to be in a state where no τ events are possible, and where the set of initial events does not meet X). Thus a process is associated to a subset of $\Sigma^* \times (\mathcal{P}(\Sigma) \cup \{\bullet\})$, where $\bullet$ represents the absence of a recorded refusal set.[3] Note that recording $\bullet$ does not imply that there is no refusal to observe, simply that we have not observed stability. The observation of the refusal $\emptyset$ implies that the process can be stable after the present trace, whereas $\bullet$ does not.

In any model $\mathcal{M}$, we say that Q *$\mathcal{M}$-refines* P, and write $P \sqsubseteq_M Q$, if the set associated to Q is a subset of that corresponding to P.

3.2 Model shifting for the failures model

The construction is as follows:

Lemma 3.1 *For each finite alphabet Σ there exists a context $\mathcal{C}$ (over an expanded alphabet) such that for any processes P and Q we have that $P \sqsubseteq_F Q$ if and only if $\mathcal{C}[P] \sqsubseteq_T \mathcal{C}[Q]$.*

Proof. Step 1: We use priority to produce a process (over an expanded alphabet) that can communicate an event x' if and only if the original process P is able to stably refuse x.

This is done by expanding the alphabet Σ to $\Sigma \cup \Sigma'$ (where Σ' contains a corresponding primed event for every event in Σ), and prioritising with respect to a partial order which prioritises each x over the corresponding x'. Recall that the definition of the priority operator means that this also causes τ to be promoted over the primed events.

We must also introduce an event *stab* to signify stability without requiring any refusals to be possible. This is necessary in order to be able to record an empty refusal set. Let the partial order $\leq_1$ be defined by $x' <_1 x \, \forall \, x \in \Sigma$, and let the context $\mathcal{C}_1$ be defined by

$$\mathcal{C}_1[P] = \text{prioritise}(P \mid\mid\mid RUN_{\Sigma' \cup \{stab\}}, \leq_1, \Sigma).$$

This process has a state ξ' for each state ξ of P, where ξ' has the same unprimed events (and corresponding transitions) as ξ. Furthermore ξ' can communicate x' just when ξ is stable and can refuse X, and *stab* just when ξ is stable.

[3] This is equivalent to the standard presentation in which a process is represented by a subset of Σ^* and one of $\Sigma^* \times \mathcal{P}(\Sigma)$: the trace component is just $\{s : (s, \bullet) \in \mathcal{P}\}$.

Step 2: We now recall that the definition of the failures model only allows a refusal set to be recorded at the *end* of a trace, and is not interested in (so does not record) what happens after the refusal set.

We gain this effect by using a regulator process to prevent a primed event (or *stab*) from being followed by an unprimed event. Let

$$\begin{aligned} \mathit{UNSTABLE} &= \Box_{x\in\Sigma} x \to \mathit{UNSTABLE} \\ &\quad \Box \Box_{x\in\Sigma'\cup\{\mathit{stab}\}} x \to \mathit{STABLE} \\ \mathit{STABLE} &= \Box_{x\in\Sigma'\cup\{\mathit{stab}\}} x \to \mathit{STABLE}, \end{aligned}$$

and define $\mathcal{C}$ by

$$\mathcal{C}[P] = \mathcal{C}_1[P] \underset{\Sigma\cup\Sigma'\cup\{\mathit{stab}\}}{\|} \mathit{UNSTABLE}.$$

A trace of $\mathcal{C}[P]$ consists of: firstly, a trace s of P; followed by, if P can after s be in a stable state, then for some such state σ_0 any string formed from the events that can be refused in σ_0, together with *stab*. The lemma clearly follows. □

It is clear that any such context must involve an operator that is not compositional over traces, for otherwise we would have $P \sqsubseteq_{\mathrm{T}} Q$ implies $\mathcal{C}[P] \sqsubseteq_{\mathrm{T}} \mathcal{C}[Q]$, which is equivalent to $P \sqsubseteq_{\mathrm{F}} Q$, and this is not true for general P and Q (consider for instance $P = a \to \mathit{STOP}$, $Q = (a \to \mathit{STOP}) \sqcap \mathit{STOP}$). It follows that only contexts which like ours involve priority can achieve this.

4 Semantic models

In order to generalise this construction to arbitrary finite observational semantic models, we must give formal definitions not only of particular models but of the very notion of a finite observational model.

4.1 Finite observations

We consider only models arising from *finite linear observations.* Intuitively, we postulate that we are able to observe the process performing a finite number of visible actions, and that where the process was stable (unable to perform a τ) immediately before an action, we are able to observe the *acceptance set* of actions it was willing to perform.

Note that we are unable to finitely observe *instability*: the most we are able to record from an action in an unstable state is that we did not *observe* stability. Thus in any context where we can observe stability we can also fail to observe it by simply not looking.

We take models to be defined over finite alphabets Σ, and take an arbitrary ordering on each finite Σ to be *alphabetical.*

The most precise finite observational model is that considering all finite linear observations, and is denoted $\mathcal{FL}$:

Definition 4.1 The set of *finite linear observations* over an alphabet Σ is

$$\mathcal{FL}_\Sigma := \{\langle A_0, a_1, A_1, \ldots, A_{n-1}, a_n, A_n\rangle : n \in \mathbb{N}, a_i \in \Sigma, A_i \subseteq \Sigma \text{ or } A_i = \bullet\},$$

where the a_i are interpreted as a sequence of communicated events, and the A_i denote stable acceptance sets, or in the case of $\bullet$ failure to observe stability. Let the set of such observations corresponding to a process P be denoted $\mathcal{FL}_\Sigma(P)$.

(Sometimes we will drop the Σ and just write $\mathcal{FL}(P)$).

More formally, $\mathcal{FL}(P)$ can be defined inductively; for instance

$$\mathcal{FL}(P \,\square\, Q) := \{\langle A \cup B\rangle\hat{}\alpha, \langle A \cup B\rangle\hat{}\beta : \langle A\rangle\hat{}\alpha \in \mathcal{FL}(P), \langle B\rangle\hat{}\beta \in \mathcal{FL}(Q)\}$$

(where $X \cup \bullet := \bullet$ for any set X). See Section 11.1.1 of [12] for further details.

Observe that $\mathcal{FL}$ has a natural partial order corresponding to extensions (where $\alpha\hat{}\langle\bullet\rangle\hat{}\beta$ and $\alpha\hat{}\langle A\rangle$ are both extended by $\alpha\hat{}\langle A\rangle\hat{}\beta$ for any set A and any α and β). Note that for any process P we have that $\mathcal{FL}(P)$ is downwards-closed with respect to this partial order.

4.2 Finite observational models

We consider precisely the models which are derivable from the observations of $\mathcal{FL}$, which are well-defined in the sense that they are compositional over CSP syntax (other than priority), and which respect extension of the alphabet Σ.

Definition 4.2 A finite observational *pre-model* $\mathcal{M}$ consists for each (finite) alphabet Σ of a set of *observations*, $\text{obs}_\Sigma(\mathcal{M})$, together with a relation $\mathcal{M}_\Sigma \subseteq \mathcal{FL}_\Sigma \times \text{obs}_\Sigma(\mathcal{M})$. The representation of a process P in $\mathcal{M}_\Sigma$ is denoted $\mathcal{M}_\Sigma(P)$, and is given by

$$\mathcal{M}_\Sigma(P) := \mathcal{M}_\Sigma(\mathcal{FL}_\Sigma(P)) = \{y \in \text{obs}_\Sigma(\mathcal{M}) : \exists\, x \in \mathcal{FL}_\Sigma(P).(x, y) \in \mathcal{M}_\Sigma\}.$$

For processes P and Q over alphabet Σ, if we have $\mathcal{M}_\Sigma(Q) \subseteq \mathcal{M}_\Sigma(P)$ then we say Q *$\mathcal{M}$-refines* P, and write $P \sqsubseteq_M Q$.

(As before we will sometimes drop the Σ).

Note that this definition is less general than if we had defined a pre-model to be any equivalence relation on $\mathcal{P}(\mathcal{FL}_\Sigma)$. For example, the equivalence relating sets of the same cardinality has no corresponding pre-model. Definition 4.2 agrees with that sketched in [12].

Without loss of generality, $\mathcal{M}_\Sigma$ does not identify any elements of $\text{obs}_\Sigma(\mathcal{M})$; that is, we have $\mathcal{M}_\Sigma^{-1}(x) = \mathcal{M}_\Sigma^{-1}(y)$ only if $x = y$ (otherwise quotient by this equivalence relation). Subject to this assumption, $\mathcal{M}_\Sigma$ induces a partial order on $\text{obs}_\Sigma(\mathcal{M})$:

Definition 4.3 The partial order *induced by $\mathcal{M}_\Sigma$ on* $\text{obs}_\Sigma(\mathcal{M})$ is given by: $x \leq y$ if and only if for all $b \in \mathcal{M}_\Sigma^{-1}(y)$ there exists $a \in \mathcal{M}_\Sigma^{-1}(x)$ with $a \leq b$.

Observe that for any process P it follows from this definition that $\mathcal{M}(P)$ is downwards-closed with respect to this partial order (since $\mathcal{FL}(P)$ is downwards-closed).

Definition 4.4 A pre-model $\mathcal{M}$ is *compositional* if for all CSP operators $\bigoplus$, say of arity k, and for all processes $P_1, \ldots, P_k$ and $Q_1, \ldots, Q_k$ such that $\mathcal{M}(P_i) = \mathcal{M}(Q_i)$

for all i, we have

$$\mathcal{M}\left(\bigoplus(P_i)_{i=1\ldots k}\right) = \mathcal{M}\left(\bigoplus(Q_i)_{i=1\ldots k}\right).$$

This means that the operator defined on processes in $\mathrm{obs}(\mathcal{M})$ by taking the pushforward of $\bigoplus$ along $\mathcal{M}$ is well-defined: for any sets $X_1, \ldots, X_k \subseteq \mathrm{obs}(\mathcal{M})$ which correspond to the images of CSP processes, take processes $P_1, \ldots, P_k$ such that $X_i = \mathcal{M}(P_i)$, and let

$$\bigoplus(X_i)_{i=1\ldots k} = \mathcal{M}\left(\bigoplus(P_i)_{i=1\ldots k}\right).$$

Definition 4.4 says that the result of this does not depend on the choice of the P_i.

Note that it is not necessary to require the equivalent of Definition 4.4 for recursion in the definition of a model, because of the following lemma which shows that least fixed point recursion is automatically well-defined (and formalises some arguments given in [12]):

Lemma 4.5 *Let $\mathcal{M}$ be a compositional pre-model. Let $\mathcal{C}_1, \mathcal{C}_2$ be CSP contexts, such that for any process P we have $\mathcal{M}(\mathcal{C}_1[P]) = \mathcal{M}(\mathcal{C}_2[P])$. Let the least fixed points of $\mathcal{C}_1$ and $\mathcal{C}_2$ (viewed as functions on $\mathcal{P}(\mathcal{FL})$ under the subset order) be P_1 and P_2 respectively. Then $\mathcal{M}(P_1) = \mathcal{M}(P_2)$.*

Proof. Using the fact that CSP contexts induce Scott-continuous functions on $\mathcal{P}(\mathcal{FL})$ (see [6], Section 2.8.2), the Kleene fixed point theorem gives that $P_i = \bigcup_{n=0}^{\infty} \mathcal{C}_i^n(\bot)$. Now any $x \in \mathcal{M}(P_1)$ is in the union taken up to some finite N, and since finite unions correspond to internal choice, and $\bot$ to the process **div**, we have that the unions up to N of $\mathcal{C}_1$ and $\mathcal{C}_2$ agree under $\mathcal{M}$ by compositionality. Hence $x \in \mathcal{M}(P_2)$, so $\mathcal{M}(P_1) \subseteq \mathcal{M}(P_2)$. Similarly $\mathcal{M}(P_2) \subseteq \mathcal{M}(P_1)$. □

Definition 4.6 A pre-model $\mathcal{M}$ is *extensional* if for all alphabets $\Sigma_1 \subseteq \Sigma_2$ we have that $\mathrm{obs}_{\Sigma_1}(\mathcal{M}) \subseteq \mathrm{obs}_{\Sigma_2}(\mathcal{M})$, and $\mathcal{M}_{\Sigma_2}$ agrees with $\mathcal{M}_{\Sigma_1}$ on $\mathcal{FL}(\Sigma_1) \times \mathrm{obs}_{\Sigma_1}(\mathcal{M})$.

Definition 4.7 A pre-model is a *model* if it is compositional and extensional.

In this setting, we now describe the five main finite observational models coarser than $\mathcal{FL}$: traces, failures, revivals, acceptances and refusal testing.

4.2.1 The traces model

The coarsest model measures only the *traces* of a process; that is, the sequences of events it is able to accept. This corresponds to the language of the process viewed as a nondeterministic finite automaton (NFA).

Definition 4.8 The *traces* model, $\mathcal{T}$, is given by

$$\mathrm{obs}_\Sigma(\mathcal{T}) = \Sigma^*, \ \mathcal{T}_\Sigma = \mathit{trace}_\Sigma$$

where *trace* is the equivalence relation which relates the observation $\langle A_0, a_1, A_1, \ldots, a_n, A_n\rangle$ to the string $a_1 \ldots a_n$.

4.2.2 Failures

The traces model gives us information about what a process is *allowed* to do, but it in some sense tells us nothing about what it is *required* to do. In particular, the process *STOP* trace-refines any other process.

In order to specify liveness properties, we can incorporate some information about the events the process is allowed to refuse, begining with the *failures* model. Intuitively, this captures traces s, together with the sets of events the process is allowed to stably refuse after s.

Definition 4.9 The *failures* model, $\mathcal{F}$, is given by

$$\mathrm{obs}_\Sigma(\mathcal{F}) = \Sigma^* \times (\mathcal{P}(\Sigma) \cup \{\bullet\}),\ \mathcal{F}_\Sigma = \mathit{fail}_\Sigma,$$

where fail_Σ relates the observation $\langle A_0, \ldots, a_n, A_n\rangle$ to all pairs $(a_1 \ldots a_n, X)$, for all $X \subseteq \Sigma \setminus A_n$ if $A_n \neq \bullet$, and for $X = \bullet$ otherwise.

4.2.3 Revivals

The next coarsest model, first introduced in [11], is the *revivals* model. Intuitively this captures traces s, together with sets X that can be stably refused after s, and events a (if any) that can then be accepted.

Definition 4.10 The *revivals* model, $\mathcal{R}$, is given by

$$\mathrm{obs}_\Sigma(\mathcal{R}) = \Sigma^* \times (\mathcal{P}(\Sigma) \cup \{\bullet\}) \times (\Sigma \cup \{\bullet\}),\ \mathcal{R}_\Sigma = \mathit{rev}_\Sigma),$$

where rev_Σ relates the observation $\langle A_0, a_1, \ldots, a_{n-1}, A_{n-1}, a_n, A_n\rangle$ to

(i) the triples $(a_1 \ldots a_{n-1}, X, a_n)$, for all $X \subseteq \Sigma \setminus A_{n-1}$ if $A_{n-1} \neq \bullet$ and for $X = \bullet$ otherwise, and

(ii) the triples $(a_1 \ldots a_n, X, \bullet)$, for all $X \subseteq \Sigma \setminus A_n$ if $A_n \neq \bullet$ and for $X = \bullet$ otherwise.

A finite linear observation is related to all triples consisting of: its initial trace; a stable refusal that could have been observed, or $\bullet$ if the original observation did not observe stability; and optionally (part (i) above) a single further event that can be accepted.

4.2.4 Acceptances

All the models considered up to now refer only to sets of refusals, which in particular are closed under subsets. The next model, *acceptances* (also known as 'ready sets'), refines the previous three and also considers the precise sets of events that can be stably accepted at the ends of traces.

Definition 4.11 The *acceptances* model, $\mathcal{A}$, is given by

$$\mathrm{obs}_\Sigma(\mathcal{A}) = \Sigma^* \times (\mathcal{P}(\Sigma) \cup \{\bullet\}),\ \mathcal{A}_\Sigma = \mathit{acc}_\Sigma,$$

where acc_Σ relates the observation $\langle A_0, a_1, \ldots, a_n, A_n\rangle$ to the pair $(a_1 \ldots a_n, A_n)$.

4.2.5 Refusal testing

The final model we consider is that of *refusal testing*, first introduced in [9]. This refines $\mathcal{F}$ and $\mathcal{R}$ by considering an entire history of events and stable refusal sets. It is incomparable to $\mathcal{A}$, because it does not capture precise acceptance sets.

Definition 4.12 The *refusal testing* model, $\mathcal{RT}$, is given by

$$\mathrm{obs}_\Sigma(\mathcal{RT}) = \{\langle X_0, a_1, X_1, \ldots, a_n, X_n\rangle : n \in \mathbb{N}, a_i \in \Sigma, X_i \subseteq \Sigma \text{ or } X_i = \bullet\}$$

$$\mathcal{RT}_\Sigma = \mathit{rt}_\Sigma,$$

where rt_Σ relates the observation $\langle A_0, \ldots, a_n, A_n \rangle$ to $\langle X_0, \ldots, a_n, X_n \rangle$, for all $X_i \subseteq \Sigma \setminus A_i$ if $A_i \neq \bullet$, and for $X_i = \bullet$ otherwise.

4.3 Rational models

We will later on wish to consider only models $\mathcal{M}$ for which the correspondence between $\mathcal{FL}$-observations and $\mathcal{M}$ observations is decidable by a finite memory computer. We will interpret this notion as saying the the relation $\mathcal{M}_\Sigma$ corresponds to the language accepted by some finite state automaton. In order to do this, we must first decide how to convert elements of $\mathcal{FL}_\Sigma$ to words in a language. We do this in the obvious way (the reasons for using fresh variables to represent the A_i will become apparent in Section 5).

Definition 4.13 The *canonical encoding* of $\mathcal{FL}_\Sigma$ is over the alphabet $\Xi := \Sigma \cup \Sigma'' \cup \mathit{Sym}$, where $\Sigma'' := \{a'' : a \in \Sigma\}$ and $\mathit{Sym} = \{\langle, \rangle, `,', \bullet\}$.[4] It is given by the representation in Definition 4.1, where sets A_i are expressed by listing the elements of Σ'' corresponding to the members of A_i in alphabetical order. We denote this encoding by $\phi_\Sigma : \mathcal{FL}_\Sigma \to \Xi^*$.

We now define a model to be *rational* (borrowing a term from automata theory) if its defining relation can be recognised (when suitably encoded) by some nondeterministic finite automaton.

Definition 4.14 A model $\mathcal{M}$ is *rational* if for every alphabet Σ, there is some finite alphabet Θ and a map $\psi_\Sigma : \mathrm{obs}_\Sigma(\mathcal{M}) \to \Theta^*$, such that there is a (nondeterministic) finite automaton $\mathcal{A}$ recognising $\{(\phi_\Sigma(x), \psi_\Sigma(y)) : (x, y) \in \mathcal{M}_\Sigma\}$, and such that ψ_Σ is *order-reflecting* (that is, $\psi_\Sigma(x) \leq \psi_\Sigma(y)$ only if $x \leq y$), with respect to the prefix partial order on Θ^*, and the partial order induced by $\mathcal{M}_\Sigma$ on $\mathrm{obs}_\Sigma(\mathcal{M})$.

What does it mean for an automaton to 'recognise' a relation?

Definition 4.15 For alphabets Σ and T, a relation $\mathcal{R} \subseteq \Sigma^* \times T^*$ is *recognised* by an automaton $\mathcal{A}$ just when:

(i) The event-set of $\mathcal{A}$ is $\mathrm{left}.\Sigma \cup \mathrm{right}.T$, and

(ii) For any $s \in \Sigma^*, t \in T^*$, we have $s\mathcal{R}t$ if and only if there is some interleaving of $\mathrm{left}.s$ and $\mathrm{right}.t$ accepted by $\mathcal{A}$.

Note that recognisability in the sense of Definition 4.15 is easily shown to be equivalent to the common notion of recognisability by a *finte state transducer* given for instance in [16], but the above definition is more convenient for our purposes. Note also that $\mathcal{FL}$ itself (viewing $\mathcal{FL}_\Sigma$ as the diagonal relation) is trivially rational.

Lemma 4.16 *The models $\mathcal{T}, \mathcal{F}, \mathcal{R}, \mathcal{A}$ and $\mathcal{RT}$ are rational.*

Proof. By inspection of Definitions 4.8–4.12. We take $\Theta = \Sigma \cup \Sigma' \cup \Sigma'' \cup \mathit{Sym}$, with Σ'' and the expression of acceptance sets as in the canonical encoding of $\mathcal{FL}$, and refusal sets expressed in the corresponding way over $\Sigma' := \{a' : a \in \Sigma\}$. □

[4] Note that this somewhat unsatisfactory notation denotes a set of four elements: the angle brackets $\langle$ and $\rangle$, the comma , and the symbol $\bullet$.

Note that not all relations are rational. For instance, the 'counting relation' mapping each finite linear observation to its length is clearly not rational. We do not know whether the additional constraint of being a finite observational model necessarily implies rationality; however, no irrational models are known. We therefore venture the following conjecture:

Conjecture 4.17 (Rationality of finite observational models) *Let $\mathcal{M}$ be a finite observational model. Then $\mathcal{M}$ is rational.*

5 Model shifting

We now come to the main substance of this paper: we prove results on 'model shifting', showing that there exist contexts allowing us to pass between different semantic models and the basic traces model. The main result is Theorem 5.4, which shows that this is possible for any rational model.

5.1 Model shifting for $\mathcal{FL}$

We begin by proving the result for the finest model, $\mathcal{FL}$. We show that there exists a context $\mathcal{C}_{\mathcal{FL}}$ such that for any process P, the finite linear observations of P correspond to the traces of $\mathcal{C}_{\mathcal{FL}}(P)$.

Lemma 5.1 (Model shifting for $\mathcal{FL}$) *For every alphabet Σ, there exists a context $\mathcal{C}_{\mathcal{FL}}$ over alphabet $T := \Sigma \cup \Sigma' \cup \Sigma'' \cup \{done\}$, and an order-reflecting map $\pi : \mathcal{FL}_\Sigma \to T^*$ (with respect to the extension partial order on $\mathcal{FL}_\Sigma$ and the prefix partial order on T^*) such that for any process P over Σ we have $\mathcal{T}(\mathcal{C}_{\mathcal{FL}}[P]) = \mathrm{pref}(\pi(\mathcal{FL}(P)))$ (where $\mathrm{pref}(X)$ is the prefix-closure of the set X).*

Proof. We will use the unprimed alphabet Σ to denote communicated events from the original trace, and the double-primed alphabet Σ'' to denote stable acceptances. Σ' will be used in an intermediate step to denote refusals, and *done* will be used to distinguish $\emptyset$ (representing an empty acceptance set) from $\bullet$ (representing a failure to observe anything).

Step 1: We first produce a process which is able to communicate events x_i', just when the original process can stably refuse the corresponding x_i. Define the partial order $\leq_1 = \langle x' <_1 x : x \in \Sigma \rangle$, which prevents refusal events when the corresponding event can occur.

Let the context $\mathcal{C}_1$ be given by

$$\mathcal{C}_1[X] = \mathrm{prioritise}(X \mid\mid\mid RUN_{\Sigma'}, \leq_1, \Sigma).$$

Note that the third argument prevents primed events from occurring in unstable states.

Step 2: We now similarly introduce acceptance events, which can happen in stable states when the corresponding refusal can't.

Similarly define the partial order $\leq_2 = \langle x'' <_2 x' : x \in \Sigma \rangle$, which prevents acceptance events when the corresponding refusal is possible. Let the context $\mathcal{C}_2$ be defined by

$$\mathcal{C}_2[X] = \mathrm{prioritise}(\mathcal{C}_1[X] \mid\mid\mid RUN_{\Sigma''}, \leq_2, \Sigma).$$

Step 3: We now ensure that an acceptance set inferred from a trace is a complete set accepted by the process under examination. This is most straightforwardly done by employing a regulator process, which can either accept an unprimed event or accept the alphabetically first refusal or acceptance event, followed by a refusal or acceptance for each event in turn. In the latter case it then communicates a *done* event, and returns to its original state.

The *done* event is necessary in order to distinguish between a terminal $\emptyset$, which can have a *done* after the last event, and a terminal $\bullet$, which cannot (observe that a $\emptyset$ cannot occur other than at the end). Finally, we hide the refusal events.

Let a and z denote the alphabetically first and last events respectively, and let $\mathrm{succ}\ x$ denote the alphabetical successor of x. Define the processes

$$\begin{aligned}
&\mathit{UNSTABLE} = \mathop{\Box}_{x \in \Sigma} x \to \mathit{UNSTABLE} \\
&\qquad\qquad\quad \Box\ a' \to \mathit{STABLE}(a) \Box\ a'' \to \mathit{STABLE}(a) \\
&\mathit{STABLE}(x) = x' \to \mathit{STABLE}(\mathrm{succ}\ x) \Box\ x'' \to \mathit{STABLE}(\mathrm{succ}\ x) \qquad (x \neq z) \\
&\mathit{STABLE}(z) = \mathit{done} \to \mathit{UNSTABLE},
\end{aligned}$$

and let

$$\mathcal{C}_{\mathcal{FL}}[X] = \left(\mathcal{C}_2[X] \mathop{\|}_{\Sigma \cup \Sigma' \cup \Sigma''} \mathit{UNSTABLE} \right) \setminus \Sigma'.$$

Step 4: We now complete the proof by defining the function π inductively as follows:

$$\begin{aligned}
\pi(s\hat{}\langle \bullet \rangle) &= \pi(s) \\
\pi(s\hat{}\langle x \rangle) &= \pi(s)\hat{}\langle x \rangle \\
\pi(s\hat{}\langle A = \{x_1, \ldots, x_k\} \rangle) &= \pi(s)\hat{}\langle x_1'' \ldots x_k'' \mathit{done} \rangle,
\end{aligned}$$

where without loss of generality the x_i are listed in alphabetical order.

It is clear that this is order-reflecting, and by the construction above satisfies $\mathcal{T}(\mathcal{C}_{\mathcal{FL}}[P]) = \mathrm{pref}(\pi(\mathcal{FL}(P)))$. □

This result allows us to translate questions of $\mathcal{FL}$-refinement into questions of trace refinement under $\mathcal{C}_{\mathcal{FL}}$, as follows:

Corollary 5.2 *For $\mathcal{C}_{\mathcal{FL}}$ as in Lemma 5.1, and for any processes P and Q, we have $P \sqsubseteq_{\mathrm{FL}} Q$ if and only if $\mathcal{C}_{\mathcal{FL}}[P] \sqsubseteq_{\mathrm{T}} \mathcal{C}_{\mathcal{FL}}[Q]$.*

Proof. Certainly if $\mathcal{FL}(Q) \subseteq \mathcal{FL}(P)$ then $\mathcal{T}(\mathcal{C}_{\mathcal{FL}}[Q]) = \mathrm{pref}(\pi(\mathcal{FL}(Q))) \subseteq \mathrm{pref}(\pi(\mathcal{FL}(P))) = \mathcal{T}(\mathcal{C}_{\mathcal{FL}}[P])$ and so $\mathcal{C}_{\mathcal{FL}}[P] \sqsubseteq_{\mathrm{T}} \mathcal{C}_{\mathcal{FL}}[Q]$.

Conversely, suppose there exists $x \in \mathcal{FL}(Q) \setminus \mathcal{FL}(P)$. Then since $\mathcal{FL}(P)$ is downwards-closed, we have $x \not\leq y$ for all $y \in \mathcal{FL}(P)$. Since π is order-reflecting, we have correspondingly $\pi(x) \not\leq \pi(y)$ for all $y \in \mathcal{FL}(P)$. Hence $\pi(x) \notin \mathrm{pref}(\pi(\mathcal{FL}(P)))$, so $\mathrm{pref}(\pi(\mathcal{FL}(Q))) \not\subseteq \mathrm{pref}(\pi(\mathcal{FL}(P)))$. □

5.2 Model shifting for rational observational models

We now have essentially all we need to prove the main theorem. We record a folk result, that any NFA can be implemented as a CSP process (up to prefix-closure, since trace-sets are prefix-closed but regular languages are not):

Lemma 5.3 (Implementation for NFA) *Let $\mathcal{A} = (\Sigma, Q, \delta, q_0, F)$ be a (non-deterministic) finite automaton. Then there exists a CSP process $P_\mathcal{A}$ such that* $\mathrm{pref}(L(\mathcal{A})) = \mathrm{pref}(\mathcal{T}(P_\mathcal{A}))$.

Proof. Trivial construction. See Chapter 7 of [10]. □

Theorem 5.4 (Model shifting for rational models) *For every rational model $\mathcal{M}$, there exists a context $\mathcal{C}_\mathcal{M}$ such that for any process P we have* $\mathcal{T}(\mathcal{C}_\mathcal{M}[P]) = \mathrm{pref}(\psi(\mathcal{M}(P)))$.

Proof. Let $\mathcal{A}$ be the automaton recognising $(\phi \times \psi)(\mathcal{M})$ (as from Definition 4.14), and let $P_\mathcal{A}$ be the corresponding process from Lemma 5.3.

We first apply Lemma 5.1 to produce a process whose traces correspond to the finite linear observations of the original process, prefixed with left: let $\mathcal{C}_{\mathcal{FL}}$ be the context from Lemma 5.1, and let the context $\mathcal{C}_1$ be defined by

$$\mathcal{C}_1[X] = \mathcal{C}_{\mathcal{FL}}[X][\![{}^{\mathrm{left}.x}/{}_x]\!].$$

We now compose in parallel with $P_\mathcal{A}$, to produde a process whose traces correspond to the $\mathcal{M}$-observations of the original process. Let $\mathcal{C}_2$ be defined by

$$\mathcal{C}_2[X] = \left(\left(\mathcal{C}_1[X] \underset{\{|\mathrm{left}|\}}{\|} P_\mathcal{A} \right) \setminus \{|\mathrm{left}|\} \right) [\![{}^{x}/{}_{\mathrm{right}.x}]\!].$$

Then the traces of $\mathcal{C}_2[X]$ are precisely the prefixes of the images under ψ of the observations corresponding to X, as required. □

By the same argument as for Corollary 5.2, we have

Corollary 5.5 *For any rational model $\mathcal{M}$, let $\mathcal{C}_\mathcal{M}$ be as in Theorem 5.4. Then for any processes P and Q, we have $P \sqsubseteq_M Q$ if and only if $\mathcal{C}_\mathcal{M}[P] \sqsubseteq_{\mathrm{T}} \mathcal{C}_\mathcal{M}[Q]$.*

6 Implementation

We demonstrate the technique by implementing contexts with the property of Corollary 5.5; source code may be found at [1].

For the sake of efficiency we work directly rather than using the general construction of Theorem 5.4. The context `C1` introduces refusal events and a `stab` event, which can occur only when the corresponding normal events can be refused. This implements the refusal testing model, and the context `CF` which allows only normal events optionally followed by some refusals (and `stab`) implements the failures model.

This is however suboptimal over large alphabets, in the typical situation where most events are refused most of the time. FDR3's inbuilt failures refinement checking is able to compare *acceptance* sets (checking that the acceptances of the specification are a subset of those of the implementation), which are typically smaller than the refusal sets.

The context `C'` introduces acceptance events which can occur only in stable states where the corresponding refusal cannot, and then blocks all refusals. The problem then is: how to check that the acceptances of the *specification* are a subset of those of the *implementation*, despite the fact that trace refinement checks for inclusion the other way?

The answer is to use priority to prevent the `stab` event from happening while acceptances are still available, so that `CFImpl'` is able to communicate only its *precise* acceptance sets. We then form `CFSpec'` by parallel composition with RUN for all the acceptance events, so that `CFSpec'` can communicate any *supersets* of its acceptance set.

Similar constructions with slightly different restrictions on the permissible sequences of events produce efficient processes for the revivals and refusal testing models. For the acceptances model, we just want to check for inclusion of the implementation's acceptance sets in those of the specification, so the context `CFImpl'` works for both the specification and the implementation; finite linear observations works similarly with failures replaced by refusal testing.

6.1 Testing

We test this implementation by constructing processes which are first distinguished by the failures, revivals, refusal testing and acceptance models respectively (the latter two being also distinguished by the finite linear observations model). The processes, and the models which do and do not distinguish them, are shown in Table 1 (recall the precision hierarchy of models: $\mathcal{T} \leq \mathcal{F} \leq \mathcal{R} \leq \{\mathcal{A}, \mathcal{RT}\} \leq \mathcal{FL}$). The correct results are obtained when these checks are run in FDR3 with the implementation described above.

Specification	Implementation	Passes	Fails
$a \to \mathbf{div}$	$a \to STOP$	$\mathcal{T}$	$\mathcal{F}$
$((a \to \mathbf{div}) \Box \mathbf{div}) \sqcap STOP$	$a \to \mathbf{div}$	$\mathcal{F}$	$\mathcal{R}$
$(a \to \mathbf{div}) \sqcap (\mathbf{div} \triangle (a \to STOP))$	$a \to STOP$	$\mathcal{R}, \mathcal{A}$	$\mathcal{RT}, \mathcal{FL}$
$(a \to STOP) \sqcap (b \to STOP)$	$(a \to STOP) \Box (b \to STOP)$	$\mathcal{R}, \mathcal{RT}$	$\mathcal{A}, \mathcal{FL}$

Table 1
Tests distinguishing levels of the model precision heirachy. $\triangle$ is the *interrupt* operator; see [12] for details.

6.2 Performance

We assess the performance of our simulation by running those examples from Table 1 of [5] which involve refinement checks (as opposed to deadlock- or divergence-freedom assertions), and comparing the timings for our construction against the time taken by FDR3's inbuilt failures refinement check (since $\mathcal{F}$ is the only model for which we have a point of comparison between a direct implementation and the methods developed in this paper). Results are shown in Table 2, for both the original and revised contexts described above; the perfomance of the $\mathcal{FL}$ check is also shown. As may be seen, performance is somewhat worse but not catastrophically so. Note however that these processes involve rather small alphabets; performance is expected to be worse for larger alphabets.

	Inbuilt $\mathcal{F}$			CF			CF'			FL		
Input File	$\lvert S\rvert$	$\lvert\Delta\rvert$	$T(s)$	$\lvert S\rvert$	$\lvert\Delta\rvert$	$T(s)$	$\lvert S\rvert$	$\lvert\Delta\rvert$	$T(s)$	$\lvert S\rvert$	$\lvert\Delta\rvert$	$T(s)$
inv	21M	220M	23	21M	220M	78	21M	220M	125	21M	220M	145
nspk	6.9M	121M	22	6.3M	114M	73	4.1M	72M	55	5.4M	97M	92
swp	24M	57M	16	30M	123M	61	43M	76M	107	42M	93M	131

Table 2
Experimental results comparing the performance of our construction with FDR3's inbuilt failures refinement check. $|S|$ is the number of states, $|\Delta|$ is the number of transitions, T is the time (in seconds), and M indicates millions.

6.3 Example: Conflict detection

We illustrate the usefulness of richer semantic models than just traces and failures by giving a sample application of the revivals model. Suppose that we have a process P consisting of the parallel composition of two sub-processes Q and R. The failures model is able to detect when P can refuse all the events of their shared alphabet, or deadlock in the case when they are synchronised on the whole alphabet. However, it is unable to distinguish between the two possible causes of this: it may be that one of the composands is able to refuse the entire shared alphabet, or it may be that each accepts some events from the shared alphabet, but the acceptances of Q and R are disjoint. We refer to the latter situation as a 'conflict'. The absence of conflict (and similar situations) is at the core of a number of useful ways of proving deadlock-freedom for networks of processes running in parallel [14].

The revivals model can be used to detect conflicts. For a process $P = Q\ {}_X\|_Y\ R$, we introduce a fresh event a to represent a generic event from the shared alphabet, and form the process $P' = Q'\ {}_{X'}\|_{Y'}\ R'$, where $Q' = Q[\![\{(x,x),(x,a) : x \in X\}]\!]$, $X' = X \cup \{a\}$, and similarly for R' and Y'. Conflicts of P now correspond to revivals $(s, X \cap Y, a)$, where s is a trace not containing a.

7 Conclusions

The result of Theorem 5.4 shows that the expressibility of all finite observational (rational) models can in some sense be simulated by the traces model using the priority operator. This provides a practical method of testing refinement over models that FDR does not directly support. While any such model could be implemented directly in the program itself, we have shown this is not necessary. This also serves to further demonstrate the power and usefulness of the priority operator (see also the previous work of the second-named author on the expressiveness of CSP with priority [13] and on 'slow abstraction' [15]).

Note that this type of construction can be used more generally. Firstly, it seems likely that the construction can be extended to non-finite models; for instance to reduce failures-divergences tests to traces-divergences, or infinite-traces-failures-divergences to infinite-traces-divergences.

Secondly, the construction does not use the requirement that a model be compositional. This means that it will work for any rational set of observable behaviours, such as the singleton failures semantics presented in [3]. The techniques described here can also be used to support the Timed Failures model of Timed CSP in FDR3 [2].

The limitation to rational models is from a theoretical point of view rather unsatisfactory, although it may be of little practical significance since all known models (and probably all models one would be likely to come up with) are clearly rational. However, Conjecture 4.17 remains of interest since a resolution in either direction would undoubtedly yield insight into the structure of the 'clouds' of models lying above $\mathcal{R}$ set out in [11].

Acknowledgements
The authors are grateful to Tom Gibson-Robinson for helpful discussions and practical assistance with FDR3. This work has been partially sponsored by DARPA under agreement number FA8750-12-2-0247.

References

[1] www.cs.ox.ac.uk/people/david.mestel/model-shifting.csp.

[2] Philip Armstrong, Gavin Lowe, Joël Ouaknine, and A.W. Roscoe. Model checking timed CSP. In Andrei Voronkov and Margarita Korovina, editors, *HOWARD-60. A Festschrift on the Occasion of Howard Barringer's 60th Birthday*, pages 13–33. EasyChair, 2014.

[3] Christie Bolton and Jim Davies. A singleton failures semantics for communicating sequential processes. *Formal Aspects of Computing*, 18(2):181–210, 2006.

[4] Thomas Gibson-Robinson, Philip Armstrong, Alexandre Boulgakov, and A.W. Roscoe. FDR3—a modern refinement checker for CSP. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 187–201. Springer, 2014.

[5] Thomas Gibson-Robinson, Henri Hansen, A.W. Roscoe, and Xu Wang. Practical partial order reduction for CSP. In *NASA Formal Methods*, pages 188–203. Springer, 2015.

[6] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1985.

[7] R. Milner. *A Calculus of Communicating Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1982.

[8] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, i. *Information and Computation*, 100(1):1–40, 1992.

[9] Iain Phillips. Refusal testing. *Theoretical Computer Science*, 50(3):241–284, 1987.

[10] A. W. Roscoe. *The Theory and Practice of Concurrency*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1997.

[11] A.W. Roscoe. Revivals, stuckness and the hierarchy of CSP models. *The Journal of Logic and Algebraic Programming*, 78(3):163–190, 2009.

[12] A.W. Roscoe. *Understanding Concurrent Systems*. Texts in Computer Science. Springer, 2010.

[13] A.W. Roscoe. The expressiveness of CSP with priority. In *Proceedings of MFPS 2015*, 2015.

[14] A.W. Roscoe and Naiem Dathi. The pursuit of deadlock freedom. *Information and Computation*, 75(3):289 – 327, 1987.

[15] A.W. Roscoe and Philippa J. Hopcroft. Slow abstraction via priority. In Zhiming Liu, Jim Woodcock, and Huibiao Zhu, editors, *Theories of Programming and Formal Methods*, pages 326–345. Springer-Verlag, Berlin, Heidelberg, 2013.

[16] J. Shallit. *A Second Course in Formal Languages and Automata Theory*. Cambridge University Press, 2009.