Math. Struct. in Comp. Science (2016), *vol.* 26, *pp.* 1022–1053.
doi:10.1017/S0960129514000334 First published online 21 November 2014

Models for CSP with availability information

GAVIN LOWE

Department of Computer Science, University of Oxford,
Wolfson Building, Parks Road, OX1 3QD, United Kingdom
Email gavin.lowe@cs.ox.ac.uk.

Received 12 March 2011; revised 1 March 2012

We consider models of CSP based on recording *availability* information, i.e. the models record what events could have been performed instead of those that were actually performed. We present many different varieties of such models. For each, we give a compositional semantics, congruent to the operational semantics, and prove full abstraction and no-junk results. We compare the expressiveness of the different models.

1. Introduction

In this paper, we consider a family of semantic models of CSP (Hoare 1985; Roscoe 1997, 2010) that record *availability* information; i.e. the models record what events could have been performed instead of those events that were actually performed. For example, the models will distinguish $a \rightarrow STOP \;\Box\; b \rightarrow STOP$ and $a \rightarrow STOP \sqcap b \rightarrow STOP$: the former offers its environment the choice between a and b, so can make a available before performing b; however, the latter decides internally whether to offer a or b, so cannot make a available before performing b.

A common way of motivating process algebras (dating back to Milner (1980)) is to view a process as a black box with which the observer interacts. The models in this paper correspond to that black box having a light for each event that turns on when the event is available (as in van Glabbeek (1993, 2001)); the observer can record which lights turn on in addition to which events are performed. The models have some similarity with the ready trace model (Olderog and Hoare 1983), in that both record availability of events; however, the latter records availability only in *stable* states, where no internal activity is possible, whereas the models of this paper record availability in *all* states.

I initially became interested in such models by considering message-passing concurrent programming languages that allow code to test whether a channel is ready for communication without actually performing the communication. For example, in JCSP (Welch *et al.* 2007), the input and output ends of channels have a method `pending()`, to test whether there is data ready to be read, or whether there is a reader ready to receive data, respectively. Similarly, Java `InputStreams` have a method `available()` that returns the number of bytes that are available to be read. Andrews (2000) gives a number of examples using such a construct.

In Lowe (2009), I considered the effect of extending CSP with a construct 'if ready a then P else Q' that tests whether the event a is ready for communication (i.e. whether this process's environment is ready to perform a), acting like P or Q

appropriately. The model in Lowe (2009) recorded what events were made available by a process, in addition to the events actually performed. That model seems rather different from standard CSP models; its study raised a number of questions. We therefore investigate models that record the availability of events more fully in this paper. We show that – even without the above construct – there are many different variations, with different expressive power.

By convention, a denotational semantic model of CSP is always *compositional*, i.e. the semantics of a composite process is given in terms of the semantics of its components. Further, there are several other desirable properties of semantic models:

Congruence to the operational semantics. The denotational semantics can either be extracted from the operational semantics, or calculated compositionally, both approaches giving the same result;

Full abstraction. The notion of semantic equivalence induced by the denotational semantics corresponds to some natural equivalence, typically defined in terms of testing;

No-junk. The denotational semantic domain corresponds precisely to the semantics of processes: for each element S of the semantic domain, we can construct a corresponding process whose semantics is S.

Each of the semantic models in this paper satisfies these properties.

In Section 2, we describe our basic model. We formalize the notion of availability of events in terms of the standard operational semantics for CSP. We then formalize the denotational semantic domain, and explain how to extract denotational information from the operational semantics. We then give a congruent compositional denotational semantics, and prove full abstraction and no-junk results.

In Section 3, we describe variations on the basic model, in two dimensions: one dimension restricts the number of observations of availability between successive standard events; the other dimension allows the simultaneous availability of multiple events to be recorded. For each resulting model, we describe compositional semantics, and full abstraction and no-junk results (we omit some of the details in the interests of brevity). We then study the relative expressive power of the models.

In Section 4, we vary the models in a different way: we record events that were available as *alternatives* to the events that were actually performed: i.e. those alternative events were available from the *same state* as the events that were performed. We again describe compositional semantics, full abstraction and no-junk results, and study the relative expressive power of the models.

Finally, in Section 5, we discuss various aspects of our models, some additional potential models, and some related work.

1.1. *Overview of CSP*

We give here a brief overview of the syntax and semantics of CSP; for simplicity and brevity, we consider a fragment of the language in this paper. We also give a brief overview of the traces and stable failures models of CSP. For more details, see Hoare (1985) and Roscoe (1997, 2010).

https://doi.org/10.1017/S0960129514000334 Published online by Cambridge University Press

CSP is a process algebra for describing programs or *processes* that interact with their environment by communication. Processes communicate via atomic events, from some set Σ. Events often involve passing values over channels; for example, the event $c.3$ represents the value 3 being passed on channel c.

The simplest process is $STOP$, which represents a deadlocked process that cannot communicate with its environment. The process div represents a divergent process that can only perform internal events.

The process $a \rightarrow P$ offers its environment the event a; if the event is performed, it then acts like P. The process $c?x \rightarrow P$ is initially willing to input a value x on channel c, i.e. it is willing to perform any event of the form $c.x$; it then acts like P (which may use x). Similarly, the process $?a : A \rightarrow P$ is initially willing to perform any event a from A; it then acts like P (which may use a).

The process $P \Box Q$ can act like either P or Q, the choice being made by the environment: the environment is offered the choice between the initial events of P and Q. By contrast, $P \sqcap Q$ can act like either P or Q, with the choice being made internally, not under the control of the environment; $\sqcap_{x:X} P_x$ nondeterministically acts like any P_x for x in X. The process $P \triangleright Q$ represents a sliding choice or timeout: it initially acts like P, but if no visible event is performed then it can internally change state to act like Q.

The process $P \ {}_A\|_B \ Q$ runs P and Q in parallel; P is restricted to performing events from the set A; Q is restricted to performing events from the set B; the two processes synchronize on events from $A \cap B$. The process $P \ ||| \ Q$ interleaves P and Q, i.e. runs them in parallel with no synchronization.

The process $P \setminus A$ acts like P, except the events from A are hidden, i.e. turned into internal, invisible events, denoted τ, which do not need to synchronize with the environment. The process $P[\![R]\!]$ represents P where events are renamed according to the relation R, i.e. $P[\![R]\!]$ can perform an event b whenever P can perform an event a such that $a \, R \, b$.

Recursive processes may be defined equationally, or using the notation $\mu X \bullet P$, which represents a process that acts like P, where each occurrence of X represents a recursive instantiation of $\mu X \bullet P$.

Prefixing ($\rightarrow$) binds tighter than each of the binary choice operators, which in turn bind tighter than the parallel operators.

CSP can be given both an operational and denotational semantics. The operational semantics is presented in Figure 1. For brevity, we omit symmetrically equivalent rules for nondeterministic choice, external choice, parallel composition and interleaving.

The denotational semantics can either be extracted from the operational semantics, or defined directly over the syntax of the language; see Roscoe (1997). It is more common to use the denotational semantics when specifying or describing the behaviours of processes, although most tools act on the operational semantics.

A *trace* of a process is a sequence of (visible) events that a process can perform. If tr is a trace, then $tr \upharpoonright A$ represents the restriction of tr to the events in A, whereas $tr \setminus A$ represents tr with the events from A removed; concatenation is written '$\frown$'; A^* represents the set of traces with events from A.

https://doi.org/10.1017/S0960129514000334 Published online by Cambridge University Press

$$?a : A \to P \xrightarrow{b} P[b/a], \text{ for } b \in A, \qquad P \sqcap Q \xrightarrow{\tau} P,$$
$$\sqcap_{i \in I} P_i \xrightarrow{\tau} P_i, \text{ for } i \in I,$$

$$\frac{P \xrightarrow{a} P'}{P \,\Box\, Q \xrightarrow{a} P'} \qquad \frac{P \xrightarrow{\tau} P'}{P \,\Box\, Q \xrightarrow{\tau} P' \,\Box\, Q}$$

$$\frac{P \xrightarrow{a} P'}{P \triangleright Q \xrightarrow{a} P'} \qquad \frac{P \xrightarrow{\tau} P'}{P \triangleright Q \xrightarrow{\tau} P' \triangleright Q}$$

$$P \triangleright Q \xrightarrow{\tau} Q$$

$$\frac{P \xrightarrow{\alpha} P'}{P \;{}_A\|_B\; Q \xrightarrow{\alpha} P' \;{}_A\|_B\; Q}\, \alpha \in A - B \cup \{\tau\} \qquad \frac{P \xrightarrow{a} P' \quad Q \xrightarrow{a} Q'}{P \;{}_A\|_B\; Q \xrightarrow{a} P' \;{}_A\|_B\; Q'}\, \alpha \in A \cap B$$

$$\frac{P \xrightarrow{\alpha} P'}{P \;|||\; Q \xrightarrow{\alpha} P' \;|||\; Q}\, \alpha \in \Sigma \cup \{\tau\}$$

$$\frac{P \xrightarrow{\alpha} P'}{P \setminus A \xrightarrow{\alpha} P' \setminus A}\, \alpha \in \Sigma - A \cup \{\tau\} \qquad \frac{P \xrightarrow{\alpha} P'}{P \setminus A \xrightarrow{\tau} P' \setminus A}\, \alpha \in A$$

$$\frac{P \xrightarrow{a} P'}{P[\![R]\!] \xrightarrow{b} P'[\![R]\!]}\, a \,R\, b \qquad \frac{P \xrightarrow{\tau} P'}{P[\![R]\!] \xrightarrow{\tau} P'[\![R]\!]}$$

$$\mu X \bullet P \xrightarrow{\tau} P[\mu X \bullet P/X].$$

Fig. 1. Operational semantics; symmetrically equivalent rules are omitted.

A *stable failure* of a process P is a pair (tr, X), which represents that P can perform the trace tr to reach a stable state (i.e. where no internal events are possible) where X can be refused, i.e. where none of the events of X is available.

2. Availability information

In this section, we consider a model that records that particular events are available during an execution. We begin by extending the operational semantics so as to formally define this notion of availability. We then define our semantic domain – traces containing both standard events and availability information – with suitable healthiness conditions. We then present compositional trace semantics, and, show that it is congruent to the operational semantics. Finally, we prove full abstraction and no-junk results: that the semantic notion of equivalence corresponds to a notion of testing equivalence, and that each element of the semantic domain corresponds to a process expressible in the syntax.

We write *offer a* to record that the event a is offered by a process, i.e. a is available. We augment the operational semantics with actions to record such offers (we term these *actions*, to distinguish them from standard events). Formally, we define a new transition relation $\longrightarrow\!\!\triangleright$ from the standard transition relation $\longrightarrow$ (as in Figure 1)

https://doi.org/10.1017/S0960129514000334 Published online by Cambridge University Press

by:

$$P \xrightarrow{\alpha}\!\!\triangleright\ Q \ \Leftrightarrow\ P \xrightarrow{\alpha} Q, \qquad \text{for } \alpha \in \Sigma^{\tau},$$
$$P \xrightarrow{\mathit{offer}\ a}\!\!\triangleright\ P \ \Leftrightarrow\ P \xrightarrow{a} .$$

Example 1.

$$a \rightarrow STOP \ \Box\ b \rightarrow STOP \xrightarrow{\mathit{offer}\ a}\!\!\triangleright\ a \rightarrow STOP \ \Box\ b \rightarrow STOP \xrightarrow{b}\!\!\triangleright\ STOP.$$

By contrast

$$a \rightarrow STOP \sqcap b \rightarrow STOP \xrightarrow{\tau}\!\!\triangleright\ b \rightarrow STOP \not\xrightarrow{\mathit{offer}\ a}\!\!\triangleright .$$

Note that the transitions corresponding to *offer* actions do not change the state of the process.

Lemma 2. The $\longrightarrow\!\!\triangleright$ relation satisfies the rules given in Figure 2. For convenience, for $A \subseteq \Sigma$, we define

$$\mathit{offer}\ A = \{\mathit{offer}\ a \mid a \in A\}, \qquad A^{\dagger} = A \cup \mathit{offer}\ A, \qquad A^{\dagger\tau} = A^{\dagger} \cup \{\tau\}.$$

We now consider an appropriate form for the denotational semantics, recording availability information. One might wonder whether it is enough to record availability information only at the *end* of a trace (by analogy to the stable failures model). However, a bit of thought shows that such a model would be equivalent to the standard traces model: a process can perform the trace $tr^{\frown}\langle \mathit{offer}\ a\rangle$ precisely if it can perform the standard trace $tr^{\frown}\langle a\rangle$.

We therefore record availability information *throughout* the trace. We define an *availability trace* to be a sequence tr in $(\Sigma^{\dagger})^{*}$. We can extract the traces (of $\Sigma^{\dagger}$ actions) from the operational semantics (following the approach in Roscoe (1997, Chapter 7)):

Definition 3. We write $P \overset{s}{\longmapsto} Q$, for $s = \langle \alpha_1, \ldots, \alpha_n\rangle \in (\Sigma^{\dagger\tau})^{*}$, if there exist $P_0 = P, P_1, \ldots, P_n = Q$ such that $P_i \xrightarrow{\alpha_{i+1}}\!\!\triangleright\ P_{i+1}$ for $i = 0, \ldots, n-1$. We write $P \overset{tr}{\Longrightarrow} Q$, for $tr \in (\Sigma^{\dagger})^{*}$, if there is some s such that $P \overset{s}{\longmapsto} Q$ and $tr = s \setminus \tau$.

Example 4. The process $a \rightarrow STOP \ \Box\ b \rightarrow STOP$ has the availability trace $\langle \mathit{offer}\ a, b\rangle$; see Example 1. However, the process $a \rightarrow STOP \sqcap b \rightarrow STOP$ does not have this trace. This model therefore distinguishes these two processes, unlike the standard traces model.

Note in particular that we may record the availability of events in unstable states (where τ events are available), by contrast with models like the stable failures model that record (un)availability information only in stable states. The following example contrasts the two models.

Example 5. The processes $a \rightarrow STOP$ and $a \rightarrow STOP \sqcap STOP$ are distinguished in the stable failures model, since the latter has stable failure $(\langle\rangle, \{a\})$; however they have the same availability traces.

https://doi.org/10.1017/S0960129514000334 Published online by Cambridge University Press

$$?a : A \to P \xrightarrow{\ offer\ b\ } ?a : A \to P, \quad \text{for } b \in A, \qquad ?a : A \to P \xrightarrow{\ b\ } P[b/a], \text{ for } b \in A,$$

$$P \sqcap Q \xrightarrow{\ \tau\ } P, \qquad \bigsqcap_{i \in I} P_i \xrightarrow{\ \tau\ } P_i, \text{ for } i \in I,$$

$$\frac{P \xrightarrow{\ a\ } P'}{P \,\Box\, Q \xrightarrow{\ a\ } P'} \qquad \frac{P \xrightarrow{\ \alpha\ } P'}{P \,\Box\, Q \xrightarrow{\ \alpha\ } P' \,\Box\, Q}\ \alpha \in \mathit{offer}\,\Sigma \cup \{\tau\}$$

$$\frac{P \xrightarrow{\ a\ } P'}{P \triangleright Q \xrightarrow{\ a\ } P'} \qquad \frac{P \xrightarrow{\ \alpha\ } P'}{P \triangleright Q \xrightarrow{\ \alpha\ } P' \triangleright Q}\ \alpha \in \mathit{offer}\,\Sigma \cup \{\tau\}$$

$$P \triangleright Q \xrightarrow{\ \tau\ } Q$$

$$\frac{P \xrightarrow{\ \alpha\ } P'}{P \;{}_A\|_B\; Q \xrightarrow{\ \alpha\ } P' \;{}_A\|_B\; Q}\ \alpha \in (A - B)^{\dagger\tau} \qquad \frac{P \xrightarrow{\ \alpha\ } P' \quad Q \xrightarrow{\ \alpha\ } Q'}{P \;{}_A\|_B\; Q \xrightarrow{\ \alpha\ } P' \;{}_A\|_B\; Q'}\ \alpha \in (A \cap B)^{\dagger}$$

$$\frac{P \xrightarrow{\ \alpha\ } P'}{P \;|||\; Q \xrightarrow{\ \alpha\ } P' \;|||\; Q}\ \alpha \in \Sigma^{\dagger\tau}$$

$$\frac{P \xrightarrow{\ \alpha\ } P'}{P \setminus A \xrightarrow{\ \alpha\ } P' \setminus A}\ \alpha \in (\Sigma - A)^{\dagger\tau} \qquad \frac{P \xrightarrow{\ \alpha\ } P'}{P \setminus A \xrightarrow{\ \tau\ } P' \setminus A}\ \alpha \in A$$

$$\frac{P \xrightarrow{\ a\ } P'}{P[\![R]\!] \xrightarrow{\ b\ } P'[\![R]\!]}\ a\,R\,b \qquad \frac{P \xrightarrow{\ \tau\ } P'}{P[\![R]\!] \xrightarrow{\ \tau\ } P'[\![R]\!]}$$

$$\frac{P \xrightarrow{\ offer\ a\ } P'}{P[\![R]\!] \xrightarrow{\ offer\ b\ } P'[\![R]\!]}\ a\,R\,b \qquad \mu X \bullet P \xrightarrow{\ \tau\ } P[\mu X \bullet P/X].$$

Fig. 2. Derived operational semantics; symmetrically equivalent rules are omitted.

The processes $(a \to STOP \triangleright b \to STOP) \sqcap (b \to STOP \triangleright a \to STOP)$ and $a \to STOP \sqcap b \to STOP$ are distinguished in the availability traces model, since only the former has the availability trace $\langle \mathit{offer}\, a, b \rangle$; however, they have the same stable failures.

We do not consider the fact that the model fails to distinguish $a \to STOP$ and $a \to STOP \sqcap STOP$ to be a problem: the two models are considering orthogonal issues, availability and stable refusal. It is straightforward to combine the two models to record the information from both; we discuss this further in the conclusions.

The availability-traces of process P are then $\{tr \mid P \overset{tr}{\Longrightarrow}\}$. The following definition captures the properties of this model.

Definition 6. The *availability traces model* $\mathscr{A}$ contains those sets $T \subseteq (\Sigma^{\dagger})^*$ that satisfy the following conditions:

1. T is non-empty and prefix-closed.
2. *offer* actions can always be remove from or duplicated within a trace:

$$tr^\frown\langle \mathit{offer}\, a\rangle^\frown tr' \in T \;\Rightarrow\; tr^\frown\langle \mathit{offer}\, a, \mathit{offer}\, a\rangle^\frown tr' \in T \wedge tr^\frown tr' \in T.$$

https://doi.org/10.1017/S0960129514000334 Published online by Cambridge University Press

3. If a process can offer an event it can perform it:

$$tr^\frown\langle \mathit{offer}\ a\rangle \in T \ \Rightarrow\ tr^\frown\langle a\rangle \in T.$$

4. If a process can perform an event it can first offer it:

$$tr^\frown\langle a\rangle^\frown tr' \in T \ \Rightarrow\ tr^\frown\langle \mathit{offer}\ a, a\rangle^\frown tr' \in T.$$

Lemma 7. For all processes P, $T = \{tr \mid P \stackrel{tr}{\Longrightarrow}\}$ is an element of the availability traces model.

Proof: (Sketch). It is enough to show that T satisfies the four healthiness conditions; each follows directly from the definitions of $\longrightarrow\!\!\triangleright$ and $\Longrightarrow$. □

Compositional traces semantics

We now give compositional rules for the traces of a process. We write $\mathit{traces}_A\ [\![P]\!]$ for the traces of P. (We include the subscript 'A' in $\mathit{traces}_A\ [\![P]\!]$ to distinguish this semantics from the standard traces semantics, $\mathit{traces}\ [\![P]\!]$.) Below we will show that these are congruent, to the operational definition above.

$STOP$ and div are equivalent in this model: neither can perform or offer events.

$$\mathit{traces}_A\ [\![STOP]\!] = \mathit{traces}_A\ [\![\mathsf{div}]\!] = \{\langle\rangle\}.$$

The process $a \rightarrow P$ can initially signal that it is offering a; it can then perform a, and continue like P.

$$\mathit{traces}_A\ [\![a \rightarrow P]\!] \ =\ \{\mathit{offer}\ a\}^* \cup \{tr^\frown\langle a\rangle^\frown tr' \mid tr \in \{\mathit{offer}\ a\}^* \wedge tr' \in \mathit{traces}_A\ [\![P]\!]\}.$$

The process $P \triangleright Q$ can either perform a trace of P, or can perform a trace of P with no standard events, and then (after the timeout) perform a trace of Q. The process $P \sqcap Q$ can perform traces of either of its components; the semantics of replicated nondeterministic choice is the obvious generalization.

$$\begin{aligned}
\mathit{traces}_A\ [\![P \triangleright Q]\!] \ &=\ \mathit{traces}_A\ [\![P]\!] \cup \\
&\quad \{tr_P{}^\frown tr_Q \mid tr_P \in \mathit{traces}_A\ [\![P]\!] \wedge tr_P \upharpoonright \Sigma = \langle\rangle \\
&\qquad\qquad \wedge\ tr_Q \in \mathit{traces}_A\ [\![Q]\!]\},\\
\mathit{traces}_A\ [\![P \sqcap Q]\!] \ &=\ \mathit{traces}_A\ [\![P]\!] \cup \mathit{traces}_A\ [\![Q]\!],\\
\mathit{traces}_A\ [\![\textstyle\bigsqcap_{i\in I} P_i]\!] \ &=\ \textstyle\bigcup_{i\in I} \mathit{traces}_A\ [\![P_i]\!].
\end{aligned}$$

Before the first visible event, the process $P \,\Box\, Q$ can perform an *offer a* action if *either* P or Q can do so. Let $tr \mid\mid\mid tr'$ be the set of ways of interleaving tr and tr' (as in Roscoe (1997, page 67)):

$$\begin{aligned}
\langle\rangle \mid\mid\mid tr' \ &=\ \{tr'\},\\
tr \mid\mid\mid \langle\rangle \ &=\ \{tr\},\\
\langle a\rangle^\frown tr \mid\mid\mid \langle b\rangle^\frown tr' \ &=\ \{\langle a\rangle^\frown tr'' \mid tr'' \in tr \mid\mid\mid \langle b\rangle^\frown tr'\} \cup \\
&\quad \{\langle b\rangle^\frown tr'' \mid tr'' \in \langle a\rangle^\frown tr \mid\mid\mid tr'\}.
\end{aligned}$$

https://doi.org/10.1017/S0960129514000334 Published online by Cambridge University Press

The three sets in the definition below correspond to the cases where (a) neither process performs any visible events, (b) P performs at least one visible event (after which, Q is turned off), and (c) the symmetric case where Q performs at least one visible event.

$$\begin{aligned}
&traces_A [\![P \,\Box\, Q]\!] = \\
&\quad \{tr \mid \exists\, tr_P \in traces_A [\![P]\!], tr_Q \in traces_A [\![Q]\!] \bullet \\
&\qquad tr_P \upharpoonright \Sigma = tr_Q \upharpoonright \Sigma = \langle\rangle \wedge tr \in tr_P \,|||\, tr_Q\} \cup \\
&\quad \{tr^\frown\langle a\rangle^\frown tr'_P \mid \exists\, tr_P{}^\frown\langle a\rangle^\frown tr'_P \in traces_A [\![P]\!], tr_Q \in traces_A [\![Q]\!] \bullet \\
&\qquad tr_P \upharpoonright \Sigma = tr_Q \upharpoonright \Sigma = \langle\rangle \wedge a \in \Sigma \wedge tr \in tr_P \,|||\, tr_Q\} \cup \\
&\quad \{tr^\frown\langle a\rangle^\frown tr'_Q \mid \exists\, tr_P \in traces_A [\![P]\!], tr_Q{}^\frown\langle a\rangle^\frown tr'_Q \in traces_A [\![Q]\!] \bullet \\
&\qquad tr_P \upharpoonright \Sigma = tr_Q \upharpoonright \Sigma = \langle\rangle \wedge a \in \Sigma \wedge tr \in tr_P \,|||\, tr_Q\}.
\end{aligned}$$

In a parallel composition of the form $P \;{}_A\|_B\; Q$, P is restricted to actions from $A^\dagger$, and Q is restricted to actions from $B^\dagger$. Further, P and Q must synchronize upon both standard events from $A \cap B$ and offers of events from $A \cap B$. We write $tr_P \underset{X}{\|} tr_Q$ for the set of ways of synchronizing tr_P and tr_Q on actions from X; this relation can be defined (as in Roscoe (1997, page 70)) by the following equations and symmetrically equivalent ones:

$$\begin{aligned}
\langle\rangle \underset{X}{\|} \langle\rangle &= \{\langle\rangle\}, \\
\langle\alpha\rangle^\frown tr \underset{X}{\|} \langle\rangle &= \{\langle\alpha\rangle^\frown tr' \mid tr' \in tr \underset{X}{\|} \langle\rangle\}, && \text{if } \alpha \notin X \\
&= \{\} && \text{if } \alpha \in X, \\
\langle\alpha\rangle^\frown tr \underset{X}{\|} \langle\beta\rangle^\frown tr' &= \{\langle\alpha\rangle^\frown tr'' \mid tr'' \in tr \underset{X}{\|} tr'\}, && \text{if } \alpha = \beta \in X, \\
&= \{\}, && \text{if } \alpha, \beta \in X,\ \alpha \neq \beta, \\
&= \{\langle\alpha\rangle^\frown tr'' \mid tr'' \in tr \underset{X}{\|} \langle\beta\rangle^\frown tr'\}, && \text{if } \alpha \notin X,\ \beta \in X, \\
&= \{\langle\alpha\rangle^\frown tr'' \mid tr'' \in tr \underset{X}{\|} \langle\beta\rangle^\frown tr'\} \cup \\
&\qquad \{\langle\beta\rangle^\frown tr'' \mid tr'' \in \langle\alpha\rangle^\frown tr \underset{X}{\|} tr'\}, && \text{if } \alpha, \beta \notin X.
\end{aligned}$$

The semantics of parallel composition is then:

$$traces_A [\![P \;{}_A\|_B\; Q]\!] = \{tr \mid \exists\, tr_P \in traces_A [\![P]\!] \cap (A^\dagger)^*, tr_Q \in traces_A [\![Q]\!] \cap (B^\dagger)^* \bullet tr \in tr_P \underset{(A\cap B)^\dagger}{\|} tr_Q\}.$$

The semantics of interleaving is similar:

$$traces_A [\![P \,|||\, Q]\!] = \{tr \mid \exists\, tr_P \in traces_A [\![P]\!], tr_Q \in traces_A [\![Q]\!] \bullet tr \in tr_P \,|||\, tr_Q\}.$$

The semantic equation for hiding of A captures that *offer* A actions are blocked, and A events are internalized.

$$traces_A [\![P \setminus A]\!] = \{tr_P \setminus A \mid tr_P \in traces_A [\![P]\!] \wedge tr_P \upharpoonright \mathit{offer}\, A = \langle\rangle\}.$$

For relational renaming, we lift the renaming to apply to *offer* actions, i.e.

$$(\mathit{offer}\, a)\, R\, (\mathit{offer}\, b) \;\text{ iff }\; a\, R\, b.$$

https://doi.org/10.1017/S0960129514000334 Published online by Cambridge University Press

We then lift the relation to traces by pointwise application. The semantic equation is then a further lift of R:

$$traces_A [\![P [\![R]\!]]\!] = \{tr \mid \exists tr_P \in traces_A [\![P]\!] \bullet tr_P \; R \; tr\}.$$

We now consider the semantics of recursion. Our approach follows the standard method using complete partial orders; see, for example, (Roscoe 1997, Appendix A.1).

Lemma 8. The availability traces model forms a complete partial order under the subset ordering $\subseteq$, with $traces_A [\![\mathsf{div}]\!] = \{\langle\rangle\}$ as the bottom element.

Proof: It is clear that $\{\langle\rangle\}$ is least under the subset ordering. It is straightforward to see that the model is closed under arbitrary unions, and hence is a complete partial order. □

The following lemma can be proved using precisely the same techniques as for the standard models (see Roscoe (1997, Section 8.2)).

Lemma 9. Each of the operators is continuous with respect to the $\subseteq$ ordering.

Hence from Tarski's fixed point theorem for complete partial orders (see, e.g. (Roscoe 1997, Theorem A.1.3)), each mapping F definable using the operators of the language has a least fixed point given by $\bigcup_{n \geqslant 0} F^n(\{\langle\rangle\})$. This justifies the following definition.

$$traces_A [\![\mu X \bullet F(X)]\!] = \text{the } \subseteq\text{-least fixed point of the semantic mapping corresponding to } F.$$

The following theorem shows that the two ways of capturing the traces are congruent.

Theorem 10. For all traces $tr \in (\Sigma^\dagger)^*$:

$$tr \in traces_A [\![P]\!] \text{ iff } P \stackrel{tr}{\Longrightarrow}.$$

Proof: This is a straightforward structural induction over the non-recursive operator, together with standard techniques for dealing with recursion (as in Roscoe (2009b, Section 5)). □

Theorem 11. For all processes, $traces_A [\![P]\!]$ is a member of the availability traces model (i.e. it satisfies the conditions of Definition 6).

Proof: This follows directly from Lemma 7 and Theorem 10. □

2.1. *Full abstraction*

We can show that this model is fully abstract with respect to a form of testing in the style of (de Nicola and Hennessy 1984). We consider tests that may detect the availability of events. Following Lowe (2009), we write $\mathsf{ready}\, a \;\&\; T$ for a test that tests whether a is available, and if so acts like the test T. We also allow a test $SUCCESS$ that represents a successful test, and a simple form of prefixing. Formally, tests are defined by the grammar:

$$T ::= SUCCESS \mid a \rightarrow T \mid \mathsf{ready}\, a \;\&\; T.$$

https://doi.org/10.1017/S0960129514000334 Published online by Cambridge University Press

We consider testing systems comprising a test T and a process P, denoted $T \parallel P$. We define the semantics of testing systems by the rules below; ω indicates that the test has succeeded, and Ω represents a terminated testing system.

$$SUCCESS \parallel P \xrightarrow{\omega} \Omega \qquad \frac{P \xrightarrow{\tau}\triangleright Q}{T \parallel P \xrightarrow{\tau} T \parallel Q}$$

$$\frac{P \xrightarrow{a}\triangleright Q}{a \to T \parallel P \xrightarrow{\tau} T \parallel Q} \qquad \frac{P \xrightarrow{offer\ a}\triangleright P}{\mathsf{ready}\ a\ \&\ T \parallel P \xrightarrow{\tau} T \parallel P}$$

We say that P may pass the test T, denoted P may T, if $T \parallel P$ can perform ω (after zero or more τs).

We now show that if two processes are denotationally different, we can produce a test to distinguish them, i.e. such that one process passes the test, and the other fails it. Let $tr \in (\Sigma^\dagger)^*$. We can construct a test T_{tr} that detects the trace tr.

$$\begin{aligned} T_{\langle\rangle} &= SUCCESS, \\ T_{\langle a\rangle^\frown tr} &= a \to T_{tr}, \\ T_{\langle offer\ a\rangle^\frown tr} &= \mathsf{ready}\ a\ \&\ T_{tr}. \end{aligned}$$

The following lemma can be proved by a straightforward induction on the length of tr:

Lemma 12. For all processes P, P may T_{tr} if and only if $tr \in traces_A [\![P]\!]$.

Theorem 13. $traces_A [\![P]\!] = traces_A [\![Q]\!]$ if and only if P and Q may pass the same tests.

Proof: The left-to-right direction is trivial. If $traces_A [\![P]\!] \neq traces_A [\![Q]\!]$ then without loss of generality suppose $tr \in traces_A [\![P]\!] - traces_A [\![Q]\!]$; then P may T_{tr} but not Q may T_{tr}. □

We now show that the model contains no junk: each element of the model corresponds to a process.

Theorem 14. Let T be a member of the availability traces model. Then there is a process P such that $traces_A [\![P]\!] = T$.

Proof: Let tr be a trace. We can construct a process P_{tr} as follows:

$$\begin{aligned} P_{\langle\rangle} &= STOP, \\ P_{\langle a\rangle^\frown tr} &= a \to P_{tr}, \\ P_{\langle offer\ a\rangle^\frown tr} &= a \to \mathsf{div} \triangleright P_{tr}. \end{aligned}$$

Then the traces of P_{tr} are just tr and those traces implied from tr by the healthiness conditions of Definition 6. Formally, we can prove this by induction on tr. For example:

— The traces of $P_{\langle a\rangle^\frown tr}$ are prefixes of traces of the form[†] $(offer\ a)^k{}^\frown\langle a\rangle^\frown tr'$, where $k \geqslant 0$ and tr' is a trace of P_{tr}. Hence (by the inductive hypothesis) tr' is implied

[†] We write $(offer\ a)^k$ to denote a trace containing k copies of $offer\ a$.

https://doi.org/10.1017/S0960129514000334 Published online by Cambridge University Press

from tr by the healthiness conditions. Thus $\langle a \rangle ^\frown tr'$ is implied from $\langle a \rangle ^\frown tr$. Finally $(\mathit{offer}\ a)^k ^\frown \langle a \rangle ^\frown tr'$ is implied from $\langle a \rangle ^\frown tr$ by k applications of healthiness condition 4.

— The traces of $P_{\langle \mathit{offer}\ a \rangle ^\frown tr}$ are of two forms:

- Prefixes of traces of the form $(\mathit{offer}\ a)^k ^\frown \langle a \rangle$, which is implied from $\langle \mathit{offer}\ a \rangle$ by healthiness conditions 2 and 3.
- Traces of the form $(\mathit{offer}\ a)^k ^\frown tr'$ where tr' is a trace of P_{tr}. Hence (by the inductive hypothesis) tr' is implied from tr by the healthiness conditions. And so $(\mathit{offer}\ a)^k ^\frown tr'$ is implied from $\langle \mathit{offer}\ a \rangle ^\frown tr$ by healthiness condition 2.

Then, given T, a member of the availability traces model, we can define $P = \bigsqcap_{tr \in T} P_{tr}$ such that $\mathit{traces}_A [\![P]\!] = T$. □

The above proof makes critical use of the timeout operator ($\triangleright$). This operator could have been replaced with hiding, since hiding can be used to simulate timeout:

$$P \triangleright Q \ = \ (P \ \Box \ \mathit{trig} \rightarrow Q) \setminus \{\mathit{trig}\}, \qquad \text{where } \mathit{trig} \text{ is a fresh event.}$$

However, without these two operators, the above result does not hold. We will need the following lemma, which can be proved by a straightforward structural induction.

Lemma 15. If P does not use the timeout or hiding operators, $P \stackrel{tr}{\longmapsto} P' \stackrel{\mathit{offer}\ a}{\longrightarrow\!\!\triangleright}$ and $P' \stackrel{\tau}{\longrightarrow\!\!\triangleright} P''$ then $P'' \stackrel{\mathit{offer}\ a}{\longrightarrow\!\!\triangleright}$, i.e. τ-transitions do not cause offers to be withdrawn.

Proposition 16. There are elements of the availability traces model that cannot be expressed without timeout or hiding.

Proof: Consider the element T of the availability traces model containing all traces of the form

$$(\mathit{offer}\ a)^j ^\frown \langle a \rangle \qquad \text{and} \qquad (\mathit{offer}\ a)^j ^\frown (\mathit{offer}\ b)^k ^\frown \langle b \rangle$$

and all prefixes. (In the full language, T corresponds to the process $a \rightarrow \mathit{STOP} \triangleright b \rightarrow \mathit{STOP}$.) Note that T does not contain the trace $\langle \mathit{offer}\ a, \mathit{offer}\ b, \mathit{offer}\ a \rangle$, so the offer of a must be withdrawn. However, *offer*-transitions do not change the state of the process, so this withdrawal must be due to a τ-transition. The above lemma then shows that T is not expressible without the timeout or hiding operators. □

3. Variations

In this section, we consider variations on the model of the previous section, extending the models along essentially two different dimensions. In Section 3.1, we consider models that place a limit on the number of *offer* actions between consecutive standard events. Then in Section 3.2, we consider models that record the availability of *sets* of events. In Section 3.3, we restrict these models to place a bound on the size of the availability sets. Finally in Section 3.4, we combine these two variations, to produce a hierarchy of different models with different expressive power (illustrated in Figure 5). For each variant, we sketch how to adapt the semantic model, and the full abstraction and no-junk results from Section 2. We concentrate on discussing the relationship between the different models.

https://doi.org/10.1017/S0960129514000334 Published online by Cambridge University Press

3.1. *Bounded availability actions*

Up to now, we have allowed arbitrarily many *offer* actions between consecutive standard events. It turns out that we can restrict this. For example, we could allow at most one *offer* action between consecutive standard events (or before the first event, or after the last event). This model is more abstract than the previous; for example, it identifies the processes

$$(a \to STOP \;\Box\; b \to STOP) \sqcap (a \to STOP \;\Box\; c \to STOP) \sqcap (b \to STOP \;\Box\; c \to STOP)$$

and

$$(a \to STOP \;\Box\; b \to STOP \;\Box\; c \to STOP),$$

whereas the previous model distinguished them by the trace $\langle \mathit{offer}\, a, \mathit{offer}\, b, c\rangle$.

More generally, we define the model that allows at most n *offer* actions between consecutive standard events. Let Obs_n be the set of availability traces with this property. Then the model $\mathscr{A}_n$ is the restriction of $\mathscr{A}$ to Obs_n, i.e. writing $traces_{A,n}$ for the semantic function for $\mathscr{A}_n$, we have $traces_{A,n}[\![P]\!] = traces_A[\![P]\!] \cap Obs_n$. In particular, $\mathscr{A}_0$ is equivalent to the standard traces model.

The following example shows that the models become strictly more refined as n increases; further, the full availability traces model $\mathscr{A}$ is finer than each of the approximations $\mathscr{A}_n$.

Example 17. Pick a set $A = \{a_0, \ldots, a_n\}$ of size $n+1$. The processes

$$\sqcap_{b \in A} ?a : A - \{b\} \to STOP \qquad \text{and} \qquad ?a : A \to STOP$$

are equivalent in model $\mathscr{A}_{n-1}$: each can perform any trace of at most $n-1$ actions of the form *offer* a_i followed by an event a_j (and prefixes of such traces). However, they are distinguished in model $\mathscr{A}_n$ and model $\mathscr{A}$ by the trace $\langle \mathit{offer}\, a_0, \mathit{offer}\, a_1, \ldots, \mathit{offer}\, a_{n-1}, a_n\rangle$.

The following alternative example uses a bounded alphabet.

Example 18. Consider the processes:

$$\begin{aligned} P_0 &= STOP, \\ P_{n+1} &= (a \to STOP \sqcap b \to STOP) \triangleright P_n. \end{aligned}$$

Suppose n is non-zero and even (the case of odd n is similar). Processes P_n and P_{n+1} can be distinguished in model $\mathscr{A}_n$ and model $\mathscr{A}$, since only P_{n+1} has the trace $\langle \mathit{offer}\, a, \mathit{offer}\, b, \mathit{offer}\, a, \mathit{offer}\, b, \ldots, \mathit{offer}\, a, \mathit{offer}\, b, a\rangle$ with n *offer* actions. However, these processes are equal in model $\mathscr{A}_{n-1}$.

Following Roscoe (2009b), we write $M \preceq M'$ if model M' is finer (i.e. distinguishes more processes) than model M, and $\prec$ for the corresponding strict relation. The above examples show

$$\mathscr{A}_0 \equiv \mathscr{T} \prec \mathscr{A}_1 \prec \mathscr{A}_2 \prec \cdots \prec \mathscr{A}.$$

It is easy to see that these models are all compositional: in all the semantic equations, the presence of a trace from Obs_n in a composite process is always implied by the

https://doi.org/10.1017/S0960129514000334 Published online by Cambridge University Press

presence of traces from Obs_n in the subcomponents. It is important, here, that the number of consecutive offers is downwards-closed: the same result would not hold if we considered a model that includes *exactly* n offer actions between successive standard events, for in an interleaving $P \mid\mid\mid Q$, a sequence of n consecutive offers may be formed from k offers of P and $n-k$ offers of Q.

In some cases, the semantic equations have to be adapted slightly to ensure the traces produced are indeed from Obs_n, for example:

$$\begin{aligned}
&traces_{A,n}\,[\![P \;{}_A\|_B\; Q]\!] = \\
&\quad \{tr \mid \exists\, tr_P \in traces_{A,n}\,[\![P]\!] \cap (A^\dagger)^*, tr_Q \in traces_{A,n}\,[\![Q]\!] \cap (B^\dagger)^* \bullet \\
&\qquad tr \in (tr_P \underset{(A\cap B)^\dagger}{\|} tr_Q) \cap Obs_n\}, \\
&traces_{A,n}\,[\![P \setminus A]\!] = \\
&\quad \{tr_P \setminus A \mid tr_P \in traces_{A,n}\,[\![P]\!] \wedge tr_P \upharpoonright offer\, A = \langle\rangle \wedge tr_P \setminus A \in Obs_n\}.
\end{aligned}$$

The healthiness conditions need to be adapted slightly to reflect that only traces from Obs_n are included. For example, condition 4 becomes

$$4'.\; tr^\frown\langle a\rangle^\frown tr' \in T \wedge tr^\frown\langle offer\, a, a\rangle^\frown tr' \in Obs_n \Rightarrow tr^\frown\langle offer\, a, a\rangle^\frown tr' \in T.$$

Finally, the full abstraction result still holds, but the tests need to be restricted to include at most n successive ready tests. And the no-junk result still holds.

3.2. *Availability sets*

The models we have considered so far have considered the availability of a *single* event at a time. If we consider the availability of a *set* of events, can we distinguish more processes? The answer turns out to be yes, but only with processes that can either diverge or exhibit unbounded nondeterminism (a result which was surprising to me).

We will consider actions of the form *offer* A, where A is a set of events, representing that all the events in A are simultaneously available. We can adapt the derived operational semantics appropriately:

$$\begin{aligned}
P \xrightarrow{\alpha}_{\mathbb{P}} Q &\;\Leftrightarrow\; P \xrightarrow{\alpha} Q, \qquad \text{for } \alpha \in \Sigma \cup \{\tau\}, \\
P \xrightarrow{offer\, A}_{\mathbb{P}} P &\;\Leftrightarrow\; \forall\, a \in A \bullet P \xrightarrow{a} .
\end{aligned}$$

Lemma 19. The $\longrightarrow_{\mathbb{P}}$ relation satisfies the rules given in Figure 3. (For standard events in $\Sigma \cup \{\tau\}$, the rules are precisely as for the $\longrightarrow$ relation, and so omitted.) For relational renaming, we lift the renaming to apply to *offer* actions, by forming the subset-closure of the relational image:

$$(offer\, A)\, R\, (offer\, B) \;\Leftrightarrow\; \forall\, b \in B \bullet \exists\, a \in A \bullet a\, R\, b.$$

For convenience, we define

$$A^{\mathbb{P}\dagger} \;=\; A \cup \{offer\, B \mid B \in \mathbb{P}A\}.$$

https://doi.org/10.1017/S0960129514000334 Published online by Cambridge University Press

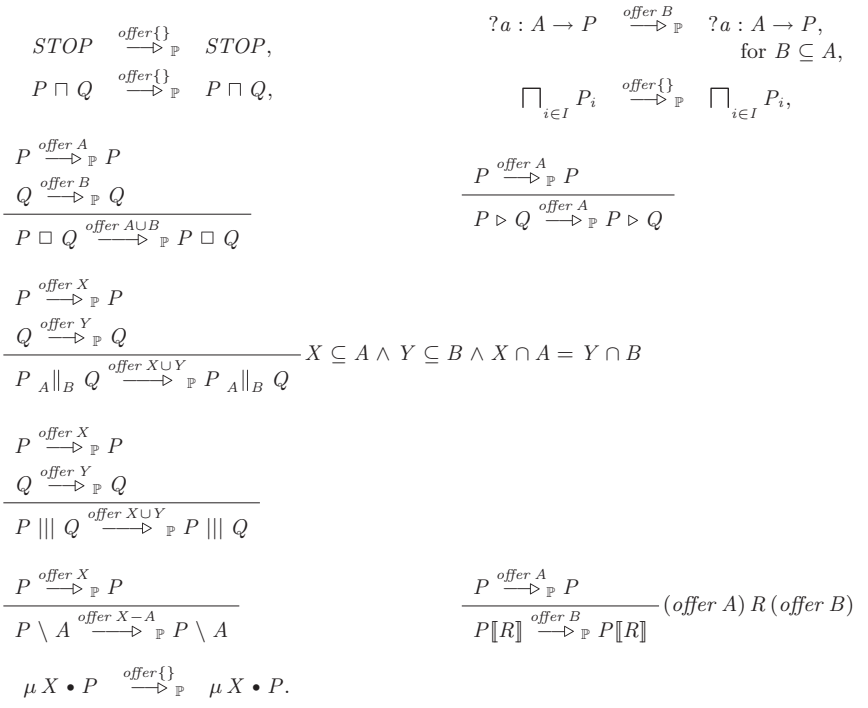

Fig. 3. Derived operational semantics for the availability sets model; symmetrically equivalent rules are omitted.

Traces will then be from $(\Sigma^{\mathbb{P}\dagger})^*$. We can extract traces of this form from the derived operational semantics as in Definition 3 (writing $\overset{tr}{\longmapsto}_{\mathbb{P}}$ and $\overset{tr}{\Longrightarrow}_{\mathbb{P}}$ for the corresponding relations).

We call this model the availability sets traces model, and will sometimes refer to the previous model as the Singleton availability traces model, in order to emphasize the difference.

Definition 20. The *availability sets traces model* $\mathscr{A}^{\mathbb{P}}$ contains those sets $T \subseteq (\Sigma^{\mathbb{P}\dagger})^*$ that satisfy the following conditions.

1. T is non-empty and prefix-closed.
2. *offer* actions can always be removed from or duplicated within a trace:

$$tr^\frown\langle \mathit{offer}\ A\rangle^\frown tr' \in T \ \Rightarrow\ tr^\frown\langle \mathit{offer}\ A, \mathit{offer}\ A\rangle^\frown tr' \in T \wedge tr^\frown tr' \in T.$$

3. If a process can offer an event it can perform it:

$$tr^\frown\langle \mathit{offer}\ A\rangle \in T \ \Rightarrow\ \forall a \in A \bullet tr^\frown\langle a\rangle \in T.$$

4. If a process can perform an event it can first offer it:

$$tr^\frown\langle a\rangle^\frown tr' \in T \ \Rightarrow\ tr^\frown\langle \mathit{offer}\{a\}, a\rangle^\frown tr' \in T.$$

https://doi.org/10.1017/S0960129514000334 Published online by Cambridge University Press

$$
\begin{array}{rcll}
\langle\rangle \underset{X}{\|}^{\mathbb{P}} \langle\rangle & = & \{\langle\rangle\}, & \\
\langle a\rangle^\frown tr \underset{X}{\|}^{\mathbb{P}} \langle\rangle & = & \{\langle a\rangle^\frown tr' \mid tr' \in tr \underset{X}{\|}^{\mathbb{P}} \langle\rangle\}, & \text{if } a \notin X \\
 & & \{\}, & \text{if } a \in X, \\
\langle \mathit{offer}\ A\rangle^\frown tr \underset{X}{\|}^{\mathbb{P}} \langle\rangle & = & \{\}, & \\
\langle a\rangle^\frown tr \underset{X}{\|}^{\mathbb{P}} \langle b\rangle^\frown tr' & = & \{\langle a\rangle^\frown tr'' \mid tr'' \in tr \underset{X}{\|}^{\mathbb{P}} tr'\}, & \text{if } a = b \in X, \\
 & & \{\}, & \text{if } a, b \in X,\ a \neq b, \\
 & & \{\langle a\rangle^\frown tr'' \mid tr'' \in tr \underset{X}{\|}^{\mathbb{P}} \langle b\rangle^\frown tr'\}, & \text{if } a \notin X,\ b \in X, \\
 & & \{\langle a\rangle^\frown tr'' \mid tr'' \in tr \underset{X}{\|}^{\mathbb{P}} \langle b\rangle^\frown tr'\} \cup & \text{if } a, b \notin X, \\
 & & \{\langle b\rangle^\frown tr'' \mid tr'' \in \langle a\rangle^\frown tr \underset{X}{\|}^{\mathbb{P}} tr'\}, & \\
\langle \mathit{offer}\ A\rangle^\frown tr \underset{X}{\|}^{\mathbb{P}} \langle \mathit{offer}\ B\rangle^\frown tr' & = & \{\mathit{offer}(A \cup B)^\frown tr'' \mid tr'' \in tr \underset{X}{\|}^{\mathbb{P}} tr'\}, & \text{if } A \cap X = B \cap X, \\
 & & \{\}, & \text{otherwise,} \\
\langle \mathit{offer}\ A\rangle^\frown tr \underset{X}{\|}^{\mathbb{P}} \langle b\rangle^\frown tr' & = & \{\}. &
\end{array}
$$

Fig. 4. Parallel combination of traces; symmetrically equivalent clauses are omitted.

5. The offers of a process are subset-closed

$$tr^\frown\langle \mathit{offer}\ A\rangle^\frown tr' \in T \wedge B \subseteq A \Rightarrow tr^\frown\langle \mathit{offer}\ B\rangle^\frown tr' \in T.$$

6. Processes can always offer the empty set

$$tr^\frown tr' \in T \Rightarrow tr^\frown\langle \mathit{offer}\{\}\rangle^\frown tr' \in T.$$

Lemma 21. For all processes P, $\{tr \mid P \overset{tr}{\Longrightarrow}_{\mathbb{P}}\}$ is an element of the availability sets traces model.

Compositional semantics. We give below semantic equations for the availability sets traces model. Most of the clauses are straightforward adaptations of the corresponding clauses in the singleton availability traces model.

For the parallel operators and external choice, we define an operator $\underset{X}{\|}^{\mathbb{P}}$ such that $tr \underset{X}{\|}^{\mathbb{P}} tr'$ gives all traces resulting from traces tr and tr', synchronizing on events and offers of events from X. The operator is defined in Figure 4.

The semantic clauses are then as follows.

$$
\begin{array}{l}
\mathit{traces}^{\mathbb{P}}_A[\![\mathit{STOP}]\!] = \mathit{traces}^{\mathbb{P}}_A[\![\mathsf{div}]\!] = (\mathit{offer}\{\})^*, \\
\mathit{traces}^{\mathbb{P}}_A[\![a \rightarrow P]\!] = \mathit{Init} \cup \{tr^\frown\langle a\rangle^\frown tr' \mid tr \in \mathit{Init} \wedge tr' \in \mathit{traces}^{\mathbb{P}}_A[\![P]\!]\}, \\
\qquad\qquad \text{where } \mathit{Init} = \{\mathit{offer}\{\}, \mathit{offer}\{a\}\}^*, \\
\mathit{traces}^{\mathbb{P}}_A[\![P \triangleright Q]\!] = \mathit{traces}^{\mathbb{P}}_A[\![P]\!] \cup \\
\qquad\qquad \{tr_P{}^\frown tr_Q \mid tr_P \in \mathit{traces}^{\mathbb{P}}_A[\![P]\!] \wedge tr_P \upharpoonright \Sigma = \langle\rangle \wedge tr_Q \in \mathit{traces}^{\mathbb{P}}_A[\![Q]\!]\}, \\
\mathit{traces}^{\mathbb{P}}_A[\![P \sqcap Q]\!] = \mathit{traces}^{\mathbb{P}}_A[\![P]\!] \cup \mathit{traces}^{\mathbb{P}}_A[\![Q]\!],
\end{array}
$$

https://doi.org/10.1017/S0960129514000334 Published online by Cambridge University Press

$$
\begin{aligned}
&\mathit{traces}_A^{\mathbb{P}}[\![P \,\Box\, Q]\!] = \\
&\quad \{tr \mid \exists\, tr_P \in \mathit{traces}_A^{\mathbb{P}}[\![P]\!], tr_Q \in \mathit{traces}_A^{\mathbb{P}}[\![Q]\!] \bullet \\
&\qquad\qquad tr_P \upharpoonright \Sigma = tr_Q \upharpoonright \Sigma = \langle\rangle \wedge tr \in tr_P \parallel^{\mathbb{P}}_{\{\}} tr_Q\} \cup \\
&\quad \{tr^\frown\langle a\rangle^\frown tr'_P \mid \exists\, tr_P{}^\frown\langle a\rangle^\frown tr'_P \in \mathit{traces}_A^{\mathbb{P}}[\![P]\!], tr_Q \in \mathit{traces}_A^{\mathbb{P}}[\![Q]\!] \bullet \\
&\qquad\qquad tr_P \upharpoonright \Sigma = tr_Q \upharpoonright \Sigma = \langle\rangle \wedge a \in \Sigma \wedge tr \in tr_P \parallel^{\mathbb{P}}_{\{\}} tr_Q\} \cup \\
&\quad \{tr^\frown\langle a\rangle^\frown tr'_Q \mid \exists\, tr_P \in \mathit{traces}_A^{\mathbb{P}}[\![P]\!], tr_Q{}^\frown\langle a\rangle^\frown tr'_Q \in \mathit{traces}_A^{\mathbb{P}}[\![Q]\!] \bullet \\
&\qquad\qquad tr_P \upharpoonright \Sigma = tr_Q \upharpoonright \Sigma = \langle\rangle \wedge a \in \Sigma \wedge tr \in tr_P \parallel^{\mathbb{P}}_{\{\}} tr_Q\},
\end{aligned}
$$

$$
\mathit{traces}_A^{\mathbb{P}}[\![P \;{}_A\|_B\; Q]\!] = \{tr \mid \exists\, tr_P \in \mathit{traces}_A^{\mathbb{P}}[\![P]\!] \cap (A^{\mathbb{P}\dagger})^*, tr_Q \in \mathit{traces}_A^{\mathbb{P}}[\![Q]\!] \cap (B^{\mathbb{P}\dagger})^* \bullet tr \in tr_P \parallel^{\mathbb{P}}_{A\cap B} tr_Q\},
$$

$$
\mathit{traces}_A^{\mathbb{P}}[\![P \mid\mid\mid Q]\!] = \{tr \mid \exists\, tr_P \in \mathit{traces}_A^{\mathbb{P}}[\![P]\!], tr_Q \in \mathit{traces}_A^{\mathbb{P}}[\![Q]\!] \bullet tr \in tr_P \parallel^{\mathbb{P}}_{\{\}} tr_Q\},
$$

$$
\mathit{traces}_A^{\mathbb{P}}[\![P \setminus A]\!] = \{tr_P \setminus A \mid tr_P \in \mathit{traces}_A^{\mathbb{P}}[\![P]\!] \wedge \forall X \bullet \mathit{offer}\, X \;\mathsf{in}\; tr_P \Rightarrow X \cap A = \{\}\},
$$

$$
\mathit{traces}_A^{\mathbb{P}}[\![P[\![R]\!]]\!] = \{tr \mid \exists\, tr_P \in \mathit{traces}_A^{\mathbb{P}}[\![P]\!] \bullet tr_P \; R \; tr\},
$$

$$
\mathit{traces}_A^{\mathbb{P}}[\![\mu X \bullet F(X)]\!] = \text{the } \subseteq\text{-least fixed point of the semantic mapping corresponding to } F.
$$

Theorem 22. The semantics is congruent to the operational semantics:

$$
tr \in \mathit{traces}_A^{\mathbb{P}}[\![P]\!] \;\text{ iff }\; P \overset{tr}{\Longrightarrow}_{\mathbb{P}}.
$$

Full abstraction and no-junk. In order to prove a full abstraction result, we extend our class of tests to include a test of the form $\mathsf{ready}\, A \,\&\, T$, which tests whether all the events in A are available, and if so acts like the test T. Formally, this test is captured by the following rule.

$$
\frac{P \overset{\mathit{offer}\, A}{\longrightarrow\!\triangleright} P}{\mathsf{ready}\, A \,\&\, T \parallel P \overset{\tau}{\longrightarrow} T \parallel P}
$$

Given $tr \in (\Sigma^{\mathbb{P}\dagger})^*$, we can construct a test T_{tr} that detects the trace tr as follows:

$$
\begin{aligned}
T_{\langle\rangle} &= \mathit{SUCCESS} \\
T_{\langle a\rangle^\frown tr} &= a \rightarrow T_{tr} \\
T_{\langle \mathit{offer}\, A\rangle^\frown tr} &= \mathsf{ready}\, A \,\&\, T_{tr}.
\end{aligned}
$$

The full abstraction proof then proceeds precisely as in Section 2.

We can prove a no-junk result as in Section 2. Given trace tr, we can construct a process P_{tr} as follows:

$$
\begin{aligned}
P_{\langle\rangle} &= \mathit{STOP}, \\
P_{\langle a\rangle^\frown tr} &= a \rightarrow P_{tr}, \\
P_{\langle \mathit{offer}\, A\rangle^\frown tr} &= (?a : A \rightarrow \mathsf{div}) \triangleright P_{tr}.
\end{aligned}
$$

https://doi.org/10.1017/S0960129514000334 Published online by Cambridge University Press

Then the traces of P_{tr} are just tr and those traces implied from tr by the healthiness conditions. Again, given an element T from the availability sets traces model, we can define $P = \bigsqcap_{tr \in T} P_{tr}$; then $traces_A^{\mathbb{P}} [\![P]\!] = T$.

Distinguishing power. We now consider the extent to which the availability sets model can distinguish processes that the singleton availability model can not.

Example 23. The availability sets traces model distinguishes the processes

$$\begin{aligned} P &= a \rightarrow STOP \,\Box\, b \rightarrow STOP, \\ Q &= (a \rightarrow STOP \sqcap b \rightarrow STOP) \triangleright Q, \end{aligned}$$

since just P has the trace $\langle \mathit{offer}\{a, b\}\rangle$. However, these are equivalent in the singleton availability traces model; in particular, both can perform arbitrary sequences of *offer a* and *offer b* actions initially.

The process Q above can diverge (i.e. perform an infinite number of internal τ events) by repeatedly performing timeouts. We can obtain a similar effect without divergence, but using unbounded nondeterminism.

Example 24. Consider

$$\begin{aligned} Q_0 &= STOP, \\ Q_{n+1} &= (a \rightarrow STOP \sqcap b \rightarrow STOP) \triangleright Q_n, \\ Q' &= \bigsqcap_{n \in \mathbb{N}} Q_n. \end{aligned}$$

Then P (from the previous example) and Q' are distinguished in the availability sets traces model but not the singleton availability traces model.

For finitely nondeterministic, non-divergent processes, it is enough to consider the availability of only *finite* sets: such a process can reach only a finite number of different states after a given trace; hence it can offer an infinite set A if and only if it can offer all its finite subsets. However, for infinitely nondeterministic processes, one can make more distinctions by considering infinite sets.

Example 25. Let A be an infinite set of events. Consider the processes

$$?a : A \rightarrow STOP \qquad \text{and} \qquad \bigsqcap_{b \in A} ?a : A - \{b\} \rightarrow STOP.$$

Then these have the same finite availability sets, but just the former has all of A available.

Theorem 26. If P and Q are non-divergent, finitely nondeterministic processes, that are equivalent in the singleton availability model, then they are equivalent in the availability sets model.

Proof: Suppose, for a contradiction, that P and Q are non-divergent and finitely nondeterministic, are equivalent in the singleton availability model, but are distinguished in the availability set model. Then, without loss of generality, there is an availability-set trace tr that is a trace of P but not of Q.

https://doi.org/10.1017/S0960129514000334 Published online by Cambridge University Press

Suppose, for the moment, that tr contains a single *offer* action, so it is of the form $tr_1 ^\frown \langle \mathit{offer}\ A\rangle ^\frown tr_2$, with no *offer* actions in tr_1 or tr_2. By the discussion above, we may assume, without loss of generality, that A is finite, say $A = \{a_1, \ldots, a_n\}$. Since Q is non-divergent and finitely nondeterministic, there is some bound, k say, on the number of consecutive τ events that it can perform after tr_1 and before the next visible event. Since P can offer all of A after tr_1, it can also offer any individual events from A, sequentially, in an arbitrary order (from clauses 2 and 5 of Definition 20). In particular, it has the singleton availability trace

$$tr_1 ^\frown \langle \mathit{offer}\ a_1, \ldots, \mathit{offer}\ a_n\rangle ^{k+1} ^\frown tr_2.$$

Since P and Q are, by assumption, equivalent in the singleton availability model, Q also has this trace. Q must perform at most k τ events within the sub-trace $\langle \mathit{offer}\ a_1, \ldots, \mathit{offer}\ a_n\rangle ^{k+1}$. This tells us that there is a sub-trace within that, of length n, containing no τ events. Within this sub-trace there are no state changes (i.e. there are only self-loops corresponding to the *offer* actions), and so all the a_i are offered in the same state. Hence $tr = tr_1 ^\frown \langle \mathit{offer}\ A\rangle ^\frown tr_2$ is an availability set trace of Q, giving a contradiction.

If tr contains multiple *offer* actions, the above argument can be applied simultaneously to all those *offer* actions. □

3.3. *Bounded sets*

We can consider some variants on the availability sets traces model.

First, let us consider the model $\mathscr{A}^k$ that places a limit of size k upon availability sets. It is reasonably straightforward to produce compositional semantics for such models: in order to deduce the offers of size at most k for a composite process, it is enough to know the offers of size at most k for the components; of course, this would not hold if we considered the offers of size exactly k. It is perhaps surprising that such a semantics is compositional, since a similar result does not hold for stable failures (Bolton and Lowe 2004) (although it is conjectured in Roscoe (2009b) that this does hold for acceptances). It is also straightforward to adapt the full abstraction and no-junk results.

Clearly, $\mathscr{A}^1 \equiv \mathscr{A}$, and $\mathscr{A}^0 \equiv \mathscr{T}$ (the standard traces model). Examples 23 and 24 show that $\mathscr{A}^2$ is finer that $\mathscr{A}^1$. We can generalize those examples to show that each model $\mathscr{A}^k$ is finer than $\mathscr{A}^{k-1}$.

Example 27. Let A_k be a set of size k. Consider

$$\begin{aligned} P_k &= ?a : A_k \to STOP, \\ Q_k &= \sqcap_{b \in A_k} (?a : A_k - \{b\} \to STOP) \triangleright Q_k. \end{aligned}$$

Then P_k and Q_k are distinguished in $\mathscr{A}^k$ since only P_k has the trace $\langle \mathit{offer}\ A_k\rangle$. However they are equivalent in $\mathscr{A}^{k-1}$: in particular, both can initially perform any trace of offers of size $k-1$.

The limit of the models $\mathscr{A}^k$ considers arbitrary *finite* availability sets; we term this $\mathscr{A}^{\mathbb{IF}}$. The model $\mathscr{A}^{\mathbb{IF}}$ distinguishes the processes P_k and Q_k from Example 27, for

https://doi.org/10.1017/S0960129514000334 Published online by Cambridge University Press

all k, so is finer than each of the models with bounded availability sets. As shown by Example 25, $\mathscr{A}^{\mathbb{F}}$ is coarser than $\mathscr{A}^{\mathbb{P}}$.

Example 27 makes critical use of timeout, as shown by the following result.

Proposition 28. Suppose P and Q do not use the timeout or hiding operators, and are equivalent in the singleton availability traces model $\mathscr{A}$. Then they are equivalent in $\mathscr{A}^{\mathbb{F}}$, and hence in all models $\mathscr{A}^k$ for finite k.

Proof: Suppose, for a contradiction, that P and Q are distinguished in $\mathscr{A}^{\mathbb{F}}$ by some trace tr that can be performed (without loss of generality) by P but not Q. Consider the singleton availability trace tr_1 obtained from tr by replacing each action of the form $offer\{a_1,\ldots,a_n\}$ by the sequence $\langle offer\ a_1,\ldots,offer\ a_n\rangle$ (where the order is arbitrary). Clearly tr_1 is a singleton availability trace of P. But P and Q are presumed equal in the singleton availability model, and hence tr_1 is also a trace of Q. However, by Lemma 15, Q cannot change state during any such subtrace $\langle offer\ a_1,\ldots,offer\ a_n\rangle$; hence Q must offer the whole of the set $\{a_1,\ldots,a_n\}$ in this state. But this means that Q has trace tr in $\mathscr{A}^{\mathbb{F}}$, giving a contradiction. □

In fact, for an arbitrary infinite cardinal κ, we can consider the model $\mathscr{A}^{\kappa}$ that places a limit of size κ upon availability sets. Example 25 showed that considering finite availability sets distinguishes fewer processes than allowing infinite availability sets, i.e. $\mathscr{A}^{\mathbb{F}} \prec \mathscr{A}^{\kappa}$. The following example shows that the models become finer as κ increases.

Example 29. Pick an infinite cardinal κ, and pick an alphabet Σ such that $card(\Sigma) \geqslant \kappa$. Then the processes

$$P_\kappa = \prod_{A\subseteq\Sigma,\, card(A)=\kappa} ?a : A \to STOP,$$
$$Q_\kappa = \prod_{A\subseteq\Sigma,\, card(A)<\kappa} ?a : A \to STOP$$

are distinguished by the model $\mathscr{A}^{\kappa}$, since only P_κ can offer sets of size κ. However, for $\lambda < \kappa$, they are not distinguished by the model $\mathscr{A}^{\lambda}$; for example, if P_κ has the trace $\langle offer\ A_1,\ldots,offer\ A_n\rangle$ in $\mathscr{A}^{\lambda}$, then $card(A_i) \leqslant \lambda < \kappa$, for each i; but also $A = \bigcup_{i=1}^{n} A_i$ has $card(A) \leqslant \lambda < \kappa$, so Q_κ can perform this trace by picking A in the nondeterministic choice.

(In fact, this example shows that these models – like the cardinals – form a proper class, rather than a set!) In most applications, the alphabet Σ is countable; these models then coincide for processes with such an alphabet. The model $\mathscr{A}^{\mathbb{P}}$ distinguishes the processes P_κ and Q_κ from Example 29, for all κ, so is finer than each of the models $\mathscr{A}^{\kappa}$.

Summarizing:

$$\mathscr{A}^0 \equiv \mathscr{T} \prec \mathscr{A}^1 \equiv \mathscr{A} \prec \mathscr{A}^2 \prec \ldots \prec \mathscr{A}^{\mathbb{F}} \prec \mathscr{A}^{\aleph_0} \prec \mathscr{A}^{\aleph_1} \prec \ldots \prec \mathscr{A}^{\mathbb{P}}.$$

3.4. *Combining the variations*

We can combine the ideas from Sections 3.1 and 3.2 to produce a family of models $\mathscr{A}^{\mathscr{K}}_{\mathscr{N}}$, where:

https://doi.org/10.1017/S0960129514000334 Published online by Cambridge University Press

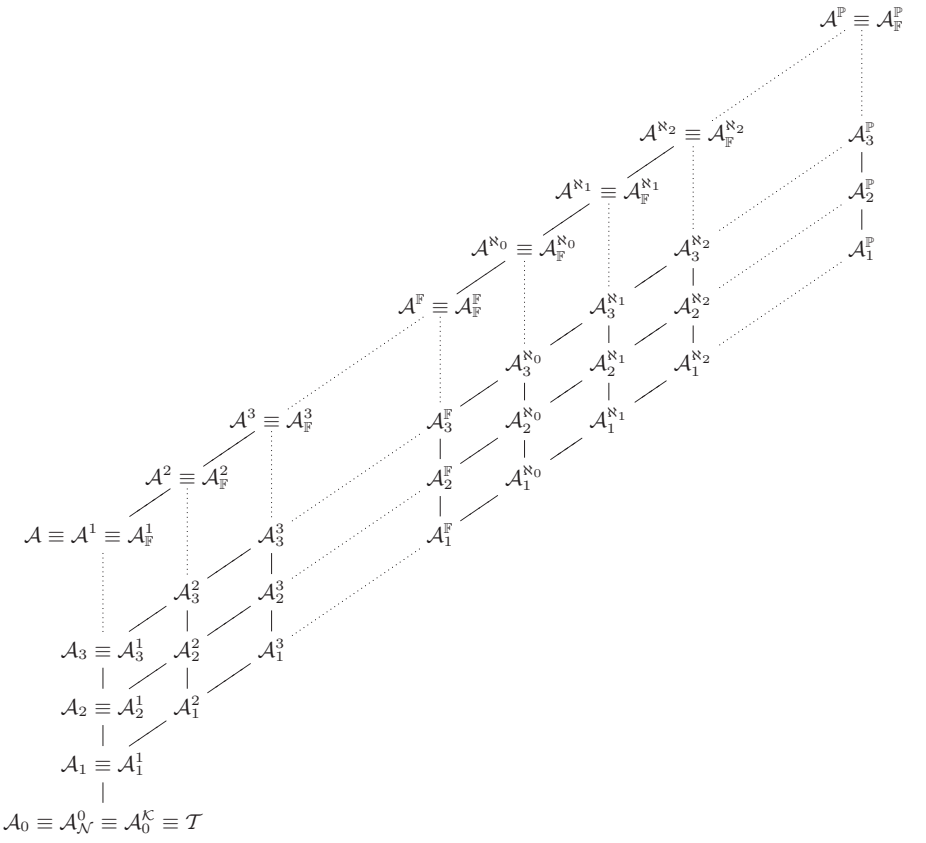

Fig. 5. The hierarchy of models.

— $\mathscr{K}$ is either a natural number k or infinite cardinal κ, indicating an upper bound on the size of availability sets, or the symbol $\mathbb{F}$ indicating arbitrary finite availability sets are allowed, or the symbol $\mathbb{P}$ indicating arbitrary availability sets are allowed;

— $\mathscr{N}$ is either a natural number n, indicating an upper bound on the number of availability sets between successive standard events, or the symbol $\mathbb{F}$ indicating any finite number is allowed.

We write $<$, $\leqslant$, etc. for the obvious orderings over these parameters.

If $\mathscr{K} = 0$ or $\mathscr{N} = 0$ then $\mathscr{A}^{\mathscr{K}}_{\mathscr{N}}$ is just the standard traces model. Further, $\mathscr{A}_{\mathscr{N}} \equiv \mathscr{A}^1_{\mathscr{N}}$ and $\mathscr{A}^{\mathscr{K}} \equiv \mathscr{A}^{\mathscr{K}}_{\mathbb{F}}$.

We can show that this family is ordered as the natural extension of the earlier (one-parameter) families; the relationship between the models is illustrated in Figure 5. In particular, these models are distinct for $\mathscr{N}, \mathscr{K} \neq 0$. We can re-use several of the earlier examples to this end. Example 18 shows that for each $\mathscr{K} \neq 0$

$$\mathscr{A}^{\mathscr{K}}_0 \prec \mathscr{A}^{\mathscr{K}}_1 \prec \mathscr{A}^{\mathscr{K}}_2 \prec \ldots \prec \mathscr{A}^{\mathscr{K}}_{\mathbb{F}}.$$

https://doi.org/10.1017/S0960129514000334 Published online by Cambridge University Press

Example 27 shows that:

$$\mathscr{A}^0_{\mathbb{F}} \prec \mathscr{A}^1_{\mathbb{F}} \prec \mathscr{A}^2_{\mathbb{F}} \prec \ldots \prec \mathscr{A}^{\mathbb{F}}_{\mathbb{F}}.$$

The following example adapts Example 27 to a bounded number of offer sets.

Example 30. Let n and k be positive natural numbers. Let A be a set of size $n \times k + 1$. Consider

$$\begin{aligned} P &= ?a : A \to STOP, \\ Q &= \sqcap_{B \subset A} ?a : B \to STOP. \end{aligned}$$

Let $A_1, \ldots, A_n, \{a\}$ be a partition of A, where each A_i is of size k. Then $\langle \mathit{offer}\ A_1, \ldots, \mathit{offer}\ A_n, a \rangle$ is a trace of P but not of Q, so these processes are distinguished by $\mathscr{A}^k_n$. However, the two processes are equivalent in $\mathscr{A}^{k-1}_n$. Hence $\mathscr{A}^{k-1}_n \prec \mathscr{A}^k_n$. Further, $\mathscr{A}^{\mathbb{F}}_n$ distinguishes these processes, for all (finite) n and k, so $\mathscr{A}^k_n \prec \mathscr{A}^{\mathbb{F}}_n$.

Example 25 shows that $\mathscr{A}^{\mathbb{F}}_{\mathscr{N}} \prec \mathscr{A}^{\kappa}_{\mathscr{N}}$ for each infinite cardinal κ and for each $\mathscr{N}$. Further, Example 29 shows that if $\lambda < \kappa$ are two infinite cardinals, then $\mathscr{A}^{\lambda}_{\mathscr{N}} \prec \mathscr{A}^{\kappa}_{\mathscr{N}} \prec \mathscr{A}^{\mathbb{P}}_{\mathscr{N}}$.

4. Alternatives

In this section, we consider models that record events that were available as *alternatives* to the events that were actually performed: i.e. those alternative events were available from the *same state* as the events that were performed. More precisely, an event b is considered an alternative to a particular transition $P \xrightarrow{a} Q$ if b was available from the state P, i.e. $P \xrightarrow{b}$.

The following example illustrates the idea.

Example 31. Consider the processes

$$\begin{aligned} P &= a \to c \to STOP \ \Box\ b \to STOP, \\ Q &= (a \to STOP \ \Box\ b \to STOP) \triangleright a \to c \to STOP. \end{aligned}$$

Then P can perform a while offering b as an alternative, and can then perform c; however, if Q performs a while offering b as an alternative, it must be from the first branch of the timeout, so can not subsequently perform c.

However, it turns out that a model that records *just* the events that were available from the *same* state as the events that were performed cannot be compositional. To see why, consider the processes

$$a \to STOP \triangleright b \to STOP \quad \text{and} \quad b \to STOP \triangleright a \to STOP.$$

These would be equivalent in such a model: each can perform a or b and offers no events as alternatives from the states where those events are performed. However,

$$a \to STOP \triangleright b \to STOP \ |||\ c \to d \to STOP$$

https://doi.org/10.1017/S0960129514000334 Published online by Cambridge University Press

can perform c while offering a as an alternative, and then perform d while offering b as an alternative; but

$$b \rightarrow STOP \triangleright a \rightarrow STOP \;|||\; c \rightarrow d \rightarrow STOP$$

cannot do this. Hence, such a model would not be compositional.

We therefore need to record *both* those alternative events that were available from the *same state* as the events that were performed, and events that were offered *before* events were performed (as in the previous models).

In order to capture this idea, we adapt the transition relation to record transitions of the form $P \xrightarrow{(alt\ B,a)}_{Alt} Q$ to indicate that P can perform a to become Q, but that P could also have performed any event from B from the same state; we specify $a \notin B$ in this case (since including a in B gives no extra information). We also record offers as in previous models. The transition relation is defined as follows:

$$\begin{aligned}
&P \xrightarrow{(alt\ B,a)}_{Alt} Q \;\; \textit{iff} \;\; P \xrightarrow{a} Q \wedge \forall b \in B \bullet P \xrightarrow{b}, \qquad \text{for } a \in \Sigma,\ a \notin B,\\
&P \xrightarrow{\tau}_{Alt} Q \;\; \textit{iff} \;\; P \xrightarrow{\tau} Q,\\
&P \xrightarrow{offer\ B}_{Alt} P \;\; \textit{iff} \;\; \forall b \in B \bullet P \xrightarrow{b}.
\end{aligned}$$

Note, in particular, that if $P \xrightarrow{a} Q$ then $P \xrightarrow{(alt\,\{\},a)}_{Alt} Q$. Note also that we do not record alternatives for τ-transitions.

Lemma 32. The $\longrightarrow_{Alt}$ relation satisfies the rules in Figure 6. (Rules for *offer* actions and the τ-promotion rules are as in Figure 3, so are omitted.) For relational renaming, we lift the renaming to alternative sets:

$$A\ R\ B \;\Leftrightarrow\; \forall b \in B \bullet \exists a \in A \bullet a\ R\ b.$$

For $A \subseteq \Sigma$, we let A^{Alt} be the set of $(alt\ B, a)$ actions and $offer\ B$ actions, where $a \in A,\ B \subseteq A,\ a \notin B$. We define an *alternatives trace* to be a sequence tr in $(\Sigma^{Alt})^*$. We can extract alternatives traces from the operational semantics, as in the earlier models (writing $\overset{tr}{\longmapsto}_{Alt}$ and $\overset{tr}{\Longrightarrow}_{Alt}$ for the corresponding relations).

Example 33. Consider again the processes

$$\begin{aligned}
P &= a \rightarrow c \rightarrow STOP \;\Box\; b \rightarrow STOP,\\
Q &= (a \rightarrow STOP \;\Box\; b \rightarrow STOP) \triangleright a \rightarrow c \rightarrow STOP.
\end{aligned}$$

Then P has the alternatives trace $\langle (alt\,\{b\}, a), (alt\,\{\}, c)\rangle$; but Q does not have this trace. Note that these two processes are equivalent in $\mathscr{A}^{\mathbb{P}}_{\mathbb{F}}$, and hence all the other availability models.

Definition 34. The *alternatives traces model* $\mathscr{A}$ contains those sets $T \subseteq (\Sigma^{Alt})^*$ such that:

1. T is non-empty and prefix-closed;
2. *offer* actions can always be remove from or duplicated within a trace:

$$tr^\frown\langle offer\ A\rangle^\frown tr' \in T \;\Rightarrow\; tr^\frown\langle offer\ A, offer\ A\rangle^\frown tr' \in T \wedge tr^\frown tr' \in T.$$

https://doi.org/10.1017/S0960129514000334 Published online by Cambridge University Press

$$?a : A \to P \quad \xrightarrow{(alt\ B, a)}_{Alt} \quad P, \text{ for } a \in A,\ B \subseteq A - \{a\},$$

$$\frac{P \xrightarrow{(alt\ X, a)}_{Alt} P' \qquad Q \xrightarrow{offer\ Y} Q}{P \,\Box\, Q \xrightarrow{(alt\ X \cup Y, a)}_{Alt} P'} a \notin Y \qquad \frac{P \xrightarrow{(alt\ X, a)}_{Alt} P'}{P \triangleright Q \xrightarrow{(alt\ X, a)}_{Alt} P'}$$

$$\frac{P \xrightarrow{(alt\ X, a)}_{Alt} P' \qquad Q \xrightarrow{offer\ Y}_{Alt} Q}{P \;{}_A\|_B\; Q \xrightarrow{(alt\ X \cup Y, a)}_{Alt} P' \;{}_A\|_B\; Q} a \in A - B \wedge a \notin Y \wedge X \subseteq A \wedge Y \subseteq B \wedge X \cap A = Y \cap B$$

$$\frac{P \xrightarrow{(alt\ X, a)}_{Alt} P' \qquad Q \xrightarrow{(alt\ Y, a)}_{Alt} Q'}{P \;{}_A\|_B\; Q \xrightarrow{(alt\ X \cup Y, a)}_{Alt} P' \;{}_A\|_B\; Q'} a \in A \cap B \wedge X \subseteq A \wedge Y \subseteq B \wedge X \cap A = Y \cap B$$

$$\frac{P \xrightarrow{(alt\ X, a)}_{Alt} P' \qquad Q \xrightarrow{offer\ Y}_{Alt} Q}{P \;|||\; Q \xrightarrow{(alt\ X \cup Y, a)}_{Alt} P' \;|||\; Q} a \notin Y$$

$$\frac{P \xrightarrow{(alt\ X, a)}_{Alt} P'}{P \setminus A \xrightarrow{(alt\ X - A, a)}_{Alt} P' \setminus A} a \notin A \qquad \frac{P \xrightarrow{(alt\ X, a)}_{Alt} P'}{P \setminus A \xrightarrow{\tau}_{Alt} P' \setminus A} a \in A$$

$$\frac{P \xrightarrow{(alt\ X, a)}_{Alt} P'}{P[\![R]\!] \xrightarrow{(alt\ Y, b)}_{Alt} P'[\![R]\!]} a\ R\ b \wedge X\ R\ Y$$

Fig. 6. Derived operational semantics for the alternatives traces model; symmetrically equivalent rules are omitted.

3. Events performed and their alternatives can be previously offered:

$$tr^\frown\langle(alt\ B, a)\rangle^\frown tr' \in T \;\Rightarrow\; tr^\frown\langle offer\ B \cup \{a\}, (alt\ B, a)\rangle^\frown tr' \in T.$$

4. If a process can offer a set of events, it can perform any member of that set with the rest available as alternatives:

$$tr^\frown\langle offer\ A\rangle \in T \;\Rightarrow\; \forall a \in A \bullet tr^\frown\langle(alt\ A - \{a\}, a)\rangle \in T.$$

5. Alternatives can be performed:

$$tr^\frown\langle(alt\ B, a)\rangle^\frown tr' \in T \;\Rightarrow\; \forall b \in B \bullet tr^\frown\langle(alt\ B - \{b\} \cup \{a\}, b)\rangle \in T.$$

6. The offers of a process are subset-closed:

$$tr^\frown\langle offer\ A\rangle^\frown tr' \in T \wedge B \subseteq A \;\Rightarrow\; tr^\frown\langle offer\ B\rangle^\frown tr' \in T.$$

7. The alternatives of a process are subset-closed:

$$tr^\frown\langle(alt\ A, a)\rangle^\frown tr' \in T \wedge B \subseteq A \;\Rightarrow\; tr^\frown\langle(alt\ B, a)\rangle^\frown tr' \in T.$$

https://doi.org/10.1017/S0960129514000334 Published online by Cambridge University Press

$$
\begin{aligned}
&\langle\rangle \underset{X}{\|}^{Alt} \langle\rangle = \{\langle\rangle\},\\
&\langle\alpha\rangle^\frown tr \underset{X}{\|}^{Alt} \langle\rangle = \{\},\\
&\langle \mathit{offer}\ A\rangle^\frown tr \underset{X}{\|}^{Alt} \langle \mathit{offer}\ B\rangle^\frown tr' =\\
&\quad \{\langle \mathit{offer}(A \cup B)\rangle^\frown tr'' \mid tr'' \in tr \underset{X}{\|}^{Alt} tr'\}, \quad \text{if } A \cap X = B \cap X,\\
&\quad \{\}, \qquad\qquad\qquad\qquad\qquad\quad \text{otherwise},\\
&\langle(\mathit{alt}\ A, a)\rangle^\frown tr \underset{X}{\|}^{Alt} \langle \mathit{offer}\ B\rangle^\frown tr' =\\
&\quad \{\langle(\mathit{alt}\ A \cup B, a)\rangle^\frown tr'' \mid tr'' \in tr \underset{X}{\|}^{Alt} tr'\}, \quad \text{if } a \notin X \wedge a \notin B \wedge A \cap X = B \cap X,\\
&\quad \{\}, \qquad\qquad\qquad\qquad\qquad\quad \text{otherwise},\\
&\langle(\mathit{alt}\ A, a)\rangle^\frown tr \underset{X}{\|}^{Alt} \langle(\mathit{alt}\ B, b)\rangle^\frown tr' =\\
&\quad \{\langle(\mathit{alt}\ A \cup B, a)\rangle^\frown tr'' \mid tr'' \in tr \underset{X}{\|}^{Alt} tr'\}, \quad \text{if } a = b \in X \wedge A \cap X = B \cap X,\\
&\quad \{\}, \qquad\qquad\qquad\qquad\qquad\quad \text{otherwise}.
\end{aligned}
$$

Fig. 7. Parallel composition of traces in the alternatives traces model; symmetrically equivalent clauses are omitted.

8. Processes can always offer the empty set:

$$tr^\frown tr' \in T \;\Rightarrow\; tr^\frown\langle \mathit{offer}\{\}\rangle^\frown tr' \in T.$$

Lemma 35. For all processes P, the set $T = \{tr \mid P \overset{tr}{\Longrightarrow}_{Alt}\}$ is an element of the alternatives traces model.

4.1. *Compositional semantics*

We now give compositional semantic equations for this model with alternatives. Most equations are straightforward adaptations of earlier equations; we discuss here a few interesting points.

For parallel composition and external choice, we define an operator $\underset{X}{\|}^{Alt}$ such that $tr \underset{X}{\|}^{Alt} tr'$ gives all traces resulting from traces tr and tr', synchronizing on actions corresponding to events from X. We arrange for each $(\mathit{alt}\ A, a)$ to synchronize with an $\mathit{offer}\ B$ action if $a \notin X$, or with a $(\mathit{alt}\ B, a)$ action if $a \in X$; in each case, A and B should contain the same events from X. Figure 7 gives the definition.

We define a hiding operator over traces such that $tr \setminus X$ removes all $(\mathit{alt}\ A, a)$ actions such that $a \in X$; the semantic equation below blocks all $\mathit{offer}\ B$ or $(\mathit{alt}\ B, a)$ events with $B \cap A \neq \{\}$.

For relational renaming, we lift the renaming to apply to alternative actions so that $(\mathit{alt}\ X, a)\ R\ (\mathit{alt}\ Y, b)$ if $a\ R\ b$ and $\forall y \in Y \bullet \exists x \in X \bullet x\ R\ y$.

The semantic equations are as follows:

$$\mathit{traces}^{Alt}\ [\![STOP]\!] = \mathit{traces}^{Alt}\ [\![\mathsf{div}]\!] = (\mathit{offer}\{\})^*,$$

https://doi.org/10.1017/S0960129514000334 Published online by Cambridge University Press

$$traces^{Alt}\ [\![a \rightarrow P]\!] =$$
$$\quad Init \cup \{tr^\frown\langle(alt\{\}, a)\rangle^\frown tr' \mid tr \in Init \wedge tr' \in traces^{Alt}\ [\![P]\!]\},$$
$$\quad \text{where } Init = \{offer\{\}, offer\{a\}\}^*,$$

$$traces^{Alt}\ [\![P \triangleright Q]\!] =$$
$$\quad traces^{Alt}\ [\![P]\!] \cup$$
$$\quad \{tr_P{}^\frown tr_Q \mid tr_P \in traces^{Alt}\ [\![P]\!] \wedge tr_P \upharpoonright \Sigma = \langle\rangle \wedge tr_Q \in traces^{Alt}\ [\![Q]\!]\},$$

$$traces^{Alt}\ [\![P \sqcap Q]\!] = traces^{Alt}\ [\![P]\!] \cup traces^{Alt}\ [\![Q]\!],$$

$$traces^{Alt}\ [\![P \ \Box\ Q]\!] =$$
$$\quad \{tr \mid \exists\, tr_P \in traces^{Alt}\ [\![P]\!], tr_Q \in traces^{Alt}\ [\![Q]\!] \bullet$$
$$\qquad tr_P \upharpoonright \Sigma = tr_Q \upharpoonright \Sigma = \langle\rangle \wedge tr \in tr_P \underset{\{\}}{\|} tr_Q\} \cup$$
$$\quad \{tr^\frown tr'_P \mid \exists\, tr_P{}^\frown\langle(alt\ B, a)\rangle^\frown tr'_P \in traces^{Alt}\ [\![P]\!], tr_Q \in traces^{Alt}\ [\![Q]\!] \bullet$$
$$\qquad tr_P \upharpoonright \Sigma = tr_Q \upharpoonright \Sigma = \langle\rangle \wedge tr \in tr_P{}^\frown\langle(alt\ B, a)\rangle \underset{\{\}}{\|} tr_Q\} \cup$$
$$\quad \{tr^\frown tr'_Q \mid \exists\, tr_P \in traces^{Alt}\ [\![P]\!], tr_Q{}^\frown\langle(alt\ B, a)\rangle^\frown tr'_Q \in traces^{Alt}\ [\![Q]\!] \bullet$$
$$\qquad tr_P \upharpoonright \Sigma = tr_Q \upharpoonright \Sigma = \langle\rangle \wedge tr \in tr_P \underset{\{\}}{\|} tr_Q{}^\frown\langle(alt\ B, a)\rangle\},$$

$$traces^{Alt}\ [\![P\ {}_A\|_B\ Q]\!] =$$
$$\quad \{tr \mid \exists\, tr_P \in traces^{Alt}\ [\![P]\!] \cap (A^{Alt})^*, tr_Q \in traces^{Alt}\ [\![Q]\!] \cap (B^{Alt})^* \bullet$$
$$\qquad tr \in tr_P \underset{A \cap B}{\|} tr_Q\},$$

$$traces^{Alt}\ [\![P \ |||\ Q]\!] =$$
$$\quad \{tr \mid \exists\, tr_P \in traces^{Alt}\ [\![P]\!], tr_Q \in traces^{Alt}\ [\![Q]\!] \bullet tr \in tr_P \underset{\{\}}{\|} tr_Q\},$$

$$traces^{Alt}\ [\![P \setminus A]\!] =$$
$$\quad \{tr_P \setminus A \mid tr_P \in traces^{Alt}\ [\![P]\!] \wedge$$
$$\qquad \forall\, B \bullet offer\ B \text{ in } tr_P \vee (alt\ B, a) \text{ in } tr_P \Rightarrow B \cap A = \{\}\},$$

$$traces^{Alt}\ [\![P[\![R]\!]]\!] = \{tr \mid \exists\, tr_P \in traces^{Alt}\ [\![P]\!] \bullet tr_P\ R\ tr\},$$

$$traces^{Alt}\ [\![\mu X \bullet F(X)]\!] =$$
$$\quad \text{the } \subseteq\text{-least fixed point of the semantic mapping corresponding to } F.$$

Theorem 36. The semantics is congruent to the operational semantics:

$$tr \in traces^{Alt}\ [\![P]\!] \text{ iff } P \overset{tr}{\Longrightarrow}_{Alt}.$$

4.2. *Full abstraction*

In order to prove a full abstraction result, we extend our class of tests to include a test of the form $(\mathsf{alt}\, A, a) \rightarrow T$, which tests whether all the events in $A \cup \{a\}$ are available, and if so performs a, and then acts like the test T. Formally, the behaviour of this test is

https://doi.org/10.1017/S0960129514000334 Published online by Cambridge University Press

captured by the following rule.

$$\frac{P \xrightarrow{(alt\ A,a)}_{Alt} Q}{(\mathsf{alt}\, A, a) \to T \parallel P \xrightarrow{\tau} T \parallel Q}$$

Given $tr \in (\Sigma^{Alt})^*$, we can construct a test T_{tr} that detects the trace tr as follows:

$$\begin{aligned} T_{\langle\rangle} &= SUCCESS, \\ T_{\langle (alt\ A,a)\rangle^\frown tr} &= (\mathsf{alt}\, A, a) \to T_{tr}, \\ T_{\langle offer\ A\rangle^\frown tr} &= \mathsf{ready}\, A \,\&\, T_{tr}. \end{aligned}$$

The full abstraction proof then proceeds precisely as in Section 2.

We can prove a no-junk result as in Section 2. Given trace tr, we can construct a process P_{tr} as follows:

$$\begin{aligned} P_{\langle\rangle} &= STOP, \\ P_{\langle (alt\ B,a)\rangle^\frown tr} &= a \to P_{tr} \,\Box\, ?b : B \to \mathsf{div}, \\ P_{\langle offer\ B\rangle^\frown tr} &= ?b : B \to \mathsf{div} \triangleright P_{tr}. \end{aligned}$$

Then the traces of P_{tr} are just tr and those traces implied from tr by the healthiness conditions. Again, given an element T from the alternatives traces model, we can define $P = \bigsqcap_{tr \in T} P_{tr}$; then $traces^{Alt}\, [\![P]\!] = T$.

4.3. *Variations*

We now consider some variations on the alternatives traces model. As with offers, we can place an upper bound $\mathscr{R}$ on the size of the alternatives set: either a natural number r, the symbol $\mathbb{F}$ indicating arbitrary finite alternatives sets are allowed, an infinite cardinal ρ, or the symbol $\mathbb{P}$ indicating arbitrary alternatives sets are allowed. We write ${}^{\mathscr{R}}\mathscr{A}^{\mathscr{K}}_{\mathscr{N}}$ for the model that has alternative information of size at most $\mathscr{R}$, and at most $\mathscr{N}$ offer sets between successive events, with offer sets of size at most $\mathscr{K}$ (where $\mathscr{N}$ and $\mathscr{K}$ are as in Section 3.4). When $\mathscr{R} = 0$, no alternatives are recorded, so ${}^{0}\mathscr{A}^{\mathscr{K}}_{\mathscr{N}} \equiv \mathscr{A}^{\mathscr{K}}_{\mathscr{N}}$.

It turns out that ${}^{\mathscr{R}}\mathscr{A}^{\mathscr{K}}_{\mathscr{N}}$ is not compositional if $\mathscr{K} < \mathscr{R}$. The following example shows this in the case that both are finite, generalizing the example at the start of this section; this example can easily be adapted to non-finite $\mathscr{K}$ or $\mathscr{R}$.

Example 37. Suppose $k < r$, with both finite. Let $\#A = \#B = r$ with $A \cap B = \{\}$. Consider,

$$\begin{aligned} P &= (?a : A \to STOP \triangleright ?b : B \to STOP) \sqcap (\textstyle\bigsqcap_{X \subseteq A \cup B, \#X = k} ?x : X \to P), \\ Q &= (?b : B \to STOP \triangleright ?a : A \to STOP) \sqcap (\textstyle\bigsqcap_{X \subseteq A \cup B, \#X = k} ?x : X \to P). \end{aligned}$$

Then P and Q are equivalent in ${}^{r}\mathscr{A}^{k}_{\mathscr{N}}$; for example each allows arbitrary offers from $A \cup B$ of size at most k, followed by an action of the form $(alt\ B - \{b\}, b)$ with $b \in B$. However, let $c, d \notin A \cup B$ and consider,

$$P \mid\mid\mid c \to d \to STOP \qquad \text{and} \qquad Q \mid\mid\mid c \to d \to STOP.$$

https://doi.org/10.1017/S0960129514000334 Published online by Cambridge University Press

Then the former has the alternatives trace $\langle (alt\ A, c), (alt\ B, d)\rangle$ but the latter does not. Hence ${}^{r}\mathscr{A}^{k}_{\mathscr{N}}$ is not compositional.

Further, if n is finite (i.e. $n \neq \mathbb{F}$) and $\mathscr{K}, \mathscr{R} \neq 0$, then ${}^{\mathscr{R}}\mathscr{A}^{\mathscr{K}}_{n}$ is not compositional, as shown by the following example.

Example 38. Let n be finite, and let $\mathscr{K}, \mathscr{R} \neq 0$. Consider

$$\begin{aligned} P_0 &= STOP, \\ P_{n+1} &= (a \rightarrow STOP \sqcap b \rightarrow STOP) \triangleright P_n. \end{aligned}$$

Then P_{n+1} and P_{n+2} are equivalent in ${}^{\mathscr{R}}\mathscr{A}^{\mathscr{K}}_{n}$: each can perform an arbitrary sequence of up to n $offer\{a\}$ or $offer\{b\}$ actions followed by either $(alt\{\}, a)$ or $(alt\{\}, b)$. However, let

$$Q = c \rightarrow d \rightarrow Q,$$

and consider $P_{n+1} \mid\mid\mid Q$ and $P_{n+2} \mid\mid\mid Q$. These are distinguished in ${}^{\mathscr{R}}\mathscr{A}^{\mathscr{K}}_{n}$ since only the latter has the trace $\langle (alt\{a\}, c), (alt\{b\}, d), (alt\{a\}, c), (alt\{b\}, d), \ldots\rangle$ of length $n+2$. Hence, ${}^{\mathscr{R}}\mathscr{A}^{\mathscr{K}}_{n}$ is not compositional.

If $\mathscr{K} \geqslant \mathscr{R}$ then the models ${}^{\mathscr{R}}\mathscr{A}^{\mathscr{K}}_{\mathbb{F}}$ are compositional. The semantic equations can be adapted appropriately to produce only suitable traces. The critical point is that an alternative of size $\mathscr{R}$ in a composite process cannot depend upon a larger offer from a component (in particular, see the case combining an offer and an alternative in Figure 7).

We now investigate the relative discriminating strengths of the models (including those that are not compositional). The following examples show that these models become more discriminating as the size of the availability parameter $\mathscr{R}$ increases.

Example 39. Let $r > 0$ be finite, $\#X = r$, and let $a \notin X$. Consider

$$\begin{aligned} P &\mathrel{\widehat{=}} a \rightarrow a \rightarrow STOP \ \Box\ ?x : X \rightarrow STOP, \\ Q &\mathrel{\widehat{=}} (a \rightarrow STOP \ \Box\ ?x : X \rightarrow STOP) \triangleright \\ &\qquad (a \rightarrow a \rightarrow STOP \ \Box \prod_{y \in X} ?x : X - \{y\} \rightarrow STOP). \end{aligned}$$

Then P and Q are distinguished in the ${}^{r}\mathscr{A}^{\mathscr{K}}_{\mathscr{N}}$ models (for all $\mathscr{K}$ and $\mathscr{N}$), since P has trace $\langle (alt\ X, a), (alt\{\}, a)\rangle$, but Q does not. However, the processes are not distinguished in model ${}^{r-1}\mathscr{A}^{\mathscr{K}}_{\mathscr{N}}$; for example, each has (for $\mathscr{K} > r$ and for each $y \in X$) the trace $\langle offer\ X \cup \{a\}, (alt\ X - \{y\}, a), (alt\{\}, a)\rangle$, with the first action coming from the first branch. Hence ${}^{r-1}\mathscr{A}^{\mathscr{K}}_{\mathscr{N}} \prec {}^{r}\mathscr{A}^{\mathscr{K}}_{\mathscr{N}}$. Further, ${}^{\mathbb{F}}\mathscr{A}^{\mathscr{K}}_{\mathscr{N}}$ distinguishes P and Q (for all finite r), and so ${}^{r}\mathscr{A}^{\mathscr{K}}_{\mathscr{N}} \prec {}^{\mathbb{F}}\mathscr{A}^{\mathscr{K}}_{\mathscr{N}}$.

Example 40. Pick an infinite cardinal ρ, and pick an alphabet Σ such that $card(\Sigma) \geqslant \rho$. Consider the following processes:

$$\begin{aligned} P &\mathrel{\widehat{=}} a \rightarrow a \rightarrow STOP \ \Box \prod_{X \subseteq \Sigma,\, card(X) = \rho} ?x : X \rightarrow STOP, \\ Q &\mathrel{\widehat{=}} (a \rightarrow STOP \ \Box \prod_{X \subseteq \Sigma,\, card(X) = \rho} ?x : X \rightarrow STOP) \triangleright \\ &\qquad (a \rightarrow a \rightarrow STOP \ \Box \prod_{X \subseteq \Sigma,\, card(X) < \rho} ?x : X \rightarrow STOP). \end{aligned}$$

https://doi.org/10.1017/S0960129514000334 Published online by Cambridge University Press

Then P and Q are distinguished in models ${}^{0}\mathscr{A}^{\mathscr{K}}_{\mathscr{N}}$ and ${}^{\mathbb{P}}\mathscr{A}^{\mathscr{K}}_{\mathscr{N}}$, since only P can offer an alternative X of size ρ in a trace of the form $\langle (alt\, X, a), (alt\, \{\}, a)\rangle$. However, they are not distinguished in models ${}^{\mathbb{F}}\mathscr{A}^{\mathscr{K}}_{\mathscr{N}}$ or ${}^{\sigma}\mathscr{A}^{\mathscr{K}}_{\mathscr{N}}$ for $\sigma < \rho$.

Summarizing the above two examples, for all $\mathscr{N}$ and $\mathscr{K}$:

$$ {}^{0}\mathscr{A}^{\mathscr{K}}_{\mathscr{N}} \prec {}^{1}\mathscr{A}^{\mathscr{K}}_{\mathscr{N}} \prec {}^{2}\mathscr{A}^{\mathscr{K}}_{\mathscr{N}} \prec \cdots \prec {}^{\mathbb{F}}\mathscr{A}^{\mathscr{K}}_{\mathscr{N}} \prec {}^{\aleph_0}\mathscr{A}^{\mathscr{K}}_{\mathscr{N}} \prec {}^{\aleph_1}\mathscr{A}^{\mathscr{K}}_{\mathscr{N}} \prec \cdots \prec {}^{\mathbb{P}}\mathscr{A}^{\mathscr{K}}_{\mathscr{N}}. $$

For a given maximum size of the alternative set, the models become more discriminating as the number of offer sets increase; Example 18 shows that for all $k \neq 0$, and for all $\mathscr{R}$,

$$ {}^{\mathscr{R}}\mathscr{A}^{k}_{0} \prec {}^{\mathscr{R}}\mathscr{A}^{k}_{1} \prec \cdots \prec {}^{\mathscr{R}}\mathscr{A}^{k}_{\mathbb{F}}. $$

We now consider how the size of offer sets affects the distinguishing power of the models.

Example 41. Let $k > 0$ be finite; let $\mathscr{N} \neq 0$; let A be a set of events with $\#A = k$; let $c \notin A$. Consider,

$$ \begin{aligned} P &= ?a : A \to STOP \triangleright c \to STOP, \\ Q &= (\textstyle\bigsqcap_{A' \subset A, \#A' = k-1} ?a : A' \to STOP \triangleright Q) \sqcap c \to STOP \sqcap ?a : A \to STOP. \end{aligned} $$

Then in ${}^{\mathscr{R}}\mathscr{A}^{k}_{\mathscr{N}}$, P and Q are distinguished since only P has trace $\langle offer\, A, (alt\, \{\}, c)\rangle$. However, they are equivalent in ${}^{\mathscr{R}}\mathscr{A}^{k-1}_{\mathscr{N}}$; for example, both have the trace $\langle (alt\, A - \{a\}, a)\rangle$ when $\mathscr{R} \geqslant k-1$ and $a \in A$; and both have traces $\langle offer\, A_1, \ldots, offer\, A_n, (alt\, \{\}, c)\rangle$ with each $A_i \subseteq A$, $\#A_i = k-1$. Further, ${}^{\mathscr{R}}\mathscr{A}^{\mathbb{F}}_{\mathscr{N}}$ distinguishes P and Q.

Example 42. Let κ be an infinite cardinal; let $\mathscr{N} \neq 0$; let A be a set of events with $\#A = \kappa$; let $c \notin A$. Consider,

$$ \begin{aligned} P &= ?a : A \to STOP \triangleright c \to STOP, \\ Q &= (\textstyle\bigsqcap_{A' \subset A, \#A' < \kappa} ?a : A' \to STOP \triangleright c \to STOP) \sqcap ?a : A \to STOP. \end{aligned} $$

Then in ${}^{\mathscr{R}}\mathscr{A}^{\kappa}_{\mathscr{N}}$, P and Q are distinguished since only P has trace $\langle offer\, A, (alt\, \{\}, c)\rangle$. However, they are equivalent in ${}^{\mathscr{R}}\mathscr{A}^{\lambda}_{\mathscr{N}}$ for $\lambda < \kappa$; for example, both have the trace $\langle (alt\, A - \{a\}, a)\rangle$ when $\mathscr{R} \geqslant \kappa$ and $a \in A$; and both have traces $\langle offer\, A_1, \ldots, offer\, A_n, (alt\, \{\}, c)\rangle$ with each $A_i \subseteq A$, $\#A_i = \lambda$. Further, ${}^{\mathscr{R}}\mathscr{A}^{\mathbb{P}}_{\mathscr{N}}$ distinguishes P and Q, but ${}^{\mathscr{R}}\mathscr{A}^{\mathbb{F}}_{\mathscr{N}}$ identifies them.

Hence, for $\mathscr{N} \neq 0$, and for all $\mathscr{R}$:

$$ {}^{\mathscr{R}}\mathscr{A}^{0}_{\mathscr{N}} \prec {}^{\mathscr{R}}\mathscr{A}^{1}_{\mathscr{N}} \prec \cdots \prec {}^{\mathscr{R}}\mathscr{A}^{\mathbb{F}}_{\mathscr{N}} \prec {}^{\mathscr{R}}\mathscr{A}^{\aleph_0}_{\mathscr{N}} \prec {}^{\mathscr{R}}\mathscr{A}^{\aleph_1}_{\mathscr{N}} \prec \cdots \prec {}^{\mathscr{R}}\mathscr{A}^{\mathbb{P}}_{\mathscr{N}}. $$

To summarize: we have shown that for each $\mathscr{R}$, the ${}^{\mathscr{R}}\mathscr{A}^{\mathscr{K}}_{\mathscr{N}}$ models (as $\mathscr{N}$ and $\mathscr{K}$ vary) form a hierarchy isomorphic to that in Figure 5, but that the models become more discriminating as $\mathscr{R}$ increases.

https://doi.org/10.1017/S0960129514000334 Published online by Cambridge University Press

5. Discussion

5.1. *Simulation and model checking*

The models described in this paper are not supported by the model checker FDR (Formal Systems Europe; Roscoe 1994). However, it is possible to simulate the semantics, using a fresh event $\texttt{offer}.A$ to simulate the action *offer* A, and a fresh event $\texttt{alt}.A.a$ to simulate the action $(alt\ A, a)$. For example, $P = a \rightarrow STOP \ \Box\ b \rightarrow STOP$ would be simulated by

$$\begin{aligned} P_{\text{sim}} &= \texttt{alt}.\{\}.a \rightarrow STOP_{\text{sim}} \ \Box\ \texttt{alt}.\{b\}.a \rightarrow STOP_{\text{sim}} \\ &\quad \Box\ \texttt{alt}.\{\}.b \rightarrow STOP_{\text{sim}} \ \Box\ \texttt{alt}.\{a\}.b \rightarrow STOP_{\text{sim}} \\ &\quad \Box\ \texttt{offer}?A : \mathbb{P}(\{a, b\}) \rightarrow P_{\text{sim}}, \\ STOP_{\text{sim}} &= \texttt{offer}.\{\} \rightarrow STOP_{\text{sim}}. \end{aligned}$$

This simulation process, then, has the same traces as the original process in the full alternatives model, but with each *offer* A or $(alt\ A, a)$ action replaced by $\texttt{offer}.A$ or $\texttt{alt}.A.a$, respectively. (Inevitably, the number of actions available in this model grows exponentially in the number of events available in standard models, which limits the size of systems that can be analysed.) The semantics in each of the other models can be obtained by restricting the size or number of `offer` and `alt` events.

In Roscoe (2009a), he shows that any operational semantics that is CSP-like, in a certain sense, can be simulated using standard CSP operators. The derived operational semantics in Figures 2, 3 and 6 are CSP-like, in this sense. Roscoe's simulation is supported by a tool by Gibson-Robinson (2010), which has been used to automate the simulation of the singleton availability model and availability sets model. This opens up the possibility of using FDR to perform analyses in these models. Note, though, that Theorem 26 shows that the singleton availability model is as expressive as the availability sets model in most practical circumstances. We see no difficulty in using the tool to also automate the alternatives model.

In practice, simulating a semantics in this way is somewhat inefficient. We are currently planning a new version of FDR, which will allow users to define their own operational semantics, which could be used to directly support models like those in the current paper.

5.2. *Related and further models*

We've explored many different models in this paper. However, we believe that there are many related models still to be explored.

In Roscoe (2009b), he investigates the hierarchy of finite linear observation models of CSP. All of these models record availability or unavailability of events only in *stable* states (if at all), unlike the models of this paper. Example 5 shows that the singleton availability model is incomparable with the stable failures model. In fact, this example shows that all of the models in this paper except the traces model are incomparable with all of the models in Roscoe's hierarchy except the traces model (so including the ready trace model (Olderog and Hoare 1983) and the refusal testing model (Mukarram 1993; Phillips 1987)); it is, perhaps, surprising that the hierarchies are so unrelated.

https://doi.org/10.1017/S0960129514000334 Published online by Cambridge University Press

We believe that we could easily adapt our models to extend any of the finite linear observation models from Roscoe (2009b), so as to obtain a hierarchy similar to that in Figure 5: in effect, the consideration of availability and alternative information is orthogonal to the finite linear observations hierarchy.

Further, we have not considered divergences within this paper. We believe that it would be straightforward to extend this work with divergences, either building models that are divergence-strict (like the traditional failures-divergences model (Hoare 1985; Roscoe 1997)), or non-divergence-strict (like the model in Roscoe (2005)).

In van Glabbeek (1993, 2001), he considers a hierarchy of different semantic models in the linear time – branching time spectrum. Several of the models correspond to standard finite linear observation models, discussed above. One other model of interest is simulation.

Definition 43 (van Glabbeek (2001)). A *simulation* is a binary relation R on processes such that for all events a, if $P\,R\,Q$ and $P \xrightarrow{a} P'$, then for some Q', $Q \xrightarrow{a} Q'$ and $P'\,R\,Q'$. Process P can be simulated by Q, denoted $P \overset{\subseteq}{\to} Q$ if there is a simulation R with $P\,R\,Q$. P and Q are *similar* if $P \overset{\subseteq}{\to} Q$ and $Q \overset{\subseteq}{\to} P$.

Proposition 44. If P and Q are similar, they are equivalent in the alternatives traces model, and hence all our other models.

Proof: If $P \overset{\subseteq}{\to} Q$ via relation R, and $P \xrightarrow{\alpha}_{Alt} P'$, then it is easy to show that $Q \xrightarrow{\alpha}_{Alt} Q'$ for some Q' such that $P'\,R\,Q'$, by a case analysis on the action α. Hence, if $P \overset{\subseteq}{\to} Q$ and $P \overset{tr}{\longmapsto}_{Alt}$, then $Q \overset{tr}{\longmapsto}_{Alt}$, by induction on the length of tr; so if $P \overset{\subseteq}{\to} Q$, then $traces^{Alt}\,[\![P]\!] \subseteq traces^{Alt}\,[\![Q]\!]$. Hence if P and Q are similar, they are equivalent in the alternatives traces model. □

The following (standard) example shows that simulation is strictly finer than all our models.

Example 45. Consider the processes

$$P = a \to (b \to c \to STOP \;\Box\; b \to d \to STOP),$$
$$Q = a \to b \to c \to STOP \;\Box\; a \to b \to d \to STOP.$$

Then $P \overset{\subseteq}{\not\to} Q$, essentially because no state of Q simulates the state $b \to c \to STOP \;\Box\; b \to d \to STOP$ of P. However, these processes are equivalent in all of our models.

We believe that there is another variant of the alternatives traces model, which considers alternatives of only the *last* event of a trace. The following example illustrates this model.

Example 46. Consider again the processes from Example 33:

$$P = a \to c \to STOP \;\Box\; b \to STOP,$$
$$Q = (a \to STOP \;\Box\; b \to STOP) \triangleright a \to c \to STOP.$$

https://doi.org/10.1017/S0960129514000334 Published online by Cambridge University Press

These processes are distinguished in the alternative traces model. However, they are equivalent when alternatives are restricted to the last event of the trace: for example, they both have traces $\langle (alt\{b\}, a)\rangle$ and $\langle offer\{a, b\}, a, (alt\{\}, c)\rangle$.

We believe that such a model would be compositional, essentially because the alternatives to the final event in a trace of a composite process depend only on the alternatives to the final event in the subcomponents. We leave further investigation of this model for future work.

Two further possible directions in which this work might be extendable would be (A) to record what events are *not* available, or (B) to record the *complete* set of events that are available. We see considerable difficulties in producing such models. To see why, consider the process $a \rightarrow P$. There are two different ways of viewing this process (which amount to different operational semantics for this process):

— One view is that the event a becomes available *immediately*. With this view: in model A, one cannot initially observe the unavailability of a; in model B, the initial complete availability set is $\{a\}$.
However, under this view, the fixed point theory does not work as required, since div is not the bottom element of the subset ordering: in model A, div has a initially unavailable; in model B, div's initial complete availability set is $\{\}$; these are both behaviours not exhibited by $a \rightarrow P$.
Further, under this view, nondeterminism is not idempotent, since, for example, $a \rightarrow P \sqcap a \rightarrow P$ has a unavailable initially; one consequence is that the proof of the no-junk result cannot be easily adapted to this view.

— The other view is that $a \rightarrow P$ takes some time to make the event a available: initially a is unavailable, but an internal state change occurs to make a available. With this view: in model A, one can initially observe the unavailability of a; in model B, the initial complete availability set is $\{\}$.
However, under this view, it turns out that the state of $a \rightarrow P$ after the a has become available cannot be expressed in the syntax of the language; this means that the proof of the no-junk result cannot be easily adapted to this view. (Proving a full abstraction result is straightforward, though.)
Further, this view leads to some common identities not holding; for example, $a \rightarrow STOP \triangleright a \rightarrow STOP$ is not the same as $a \rightarrow STOP$, since the former has a trace where a is available and then unavailable.

Similar problems arise if one tries to record (A′) what events are *not* available as alternatives, or (B′) the *complete* set of alternative events.

As noted in the introduction, in Lowe (2009) we considered models for an extended version of CSP with a construct 'if ready a then P else Q'. This construct tests whether or not its environment offers a, so the model has much in common with model A above (and was built following the second view). As such, it did not have a no-junk result. Further, it did not have a full abstraction result, since it distinguished if ready a then P else P and P, but no reasonable test would distinguish these processes.

https://doi.org/10.1017/S0960129514000334 Published online by Cambridge University Press

Acknowledgements

I would like to thank Bill Roscoe, Tom Gibson–Robinson and the anonymous referees for useful comments on this work.

References

Andrews, G. R. (2000) *Foundations of Multithreaded, Parallel, and Distributed Programming,* Addison-Wesley.

Bolton, C. and Lowe, G. (2004) A hierarchy of failures-based models. *Electronic Notes in Theoretical Computer Science* **96** 129–152.

de Nicola, R. and Hennessy, M. C. B. (1984) Testing equivalences for processes. *Theoretical Computer Science* **34** 83–133.

Formal Systems (Europe) Ltd. (2005) *Failures Divergence Refinement – User Manual and Tutorial*, Version 2.8.2.

Gibson-Robinson, T. (2010) *Tyger: A Tool for Automatically Simulating CSP-Like Languages in CSP,* Master's thesis, Oxford University `http://www.cs.ox.ac.uk/files/4607/Thesis.pdf`.

Hoare, C. A. R. (1985) *Communicating Sequential Processes*, Prentice Hall.

Lowe, G. (2009) Extending CSP with tests for availability. In: *Proceedings of Concurrent Process Architectures*, IOS Press 325–348.

Milner, R. (1980) *A Calculus of Communicating Systems. Springer Lecture Notes in Computer Science* **92**.

Mukarram, A. (1993) *A Refusal Testing Model for CSP*, D. Phil thesis, Oxford.

Olderog, E. R. and Hoare, C. A. R. (1983) Specification-oriented semantics for communicating processes. In: Diaz, J. (ed.) 10th ICALP. *Lecture Notes in Computer Science* **154** 561–572.

Phillips, I. (1987) Refusal testing. *Theoretical Computer Science.*

Roscoe, A. W. (1994) Model-checking CSP. In: *A Classical Mind, Essays in Honour of C. A. R. Hoare*, Prentice-Hall 353–378.

Roscoe, A. W. (1997) *The Theory and Practice of Concurrency*, Prentice Hall.

Roscoe, A. W. (2005) Seeing beyond divergence. In: Proceedings of '25 Years of CSP'. *Lecture Notes in Computer Science* **3525** 15–25.

Roscoe, A. W. (2009) On the expressiveness of CSP. Available via `http://web.comlab.ox.ac.uk//files/1383/complete(3).pdf`.

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

Roscoe, A. W. (2010) *Understanding Concurrent Systems*, Springer.

van Glabbeek, R. J. (1993) The linear time–branching time spectrum II; the semantics of sequential systems with silent moves (extended abstract). In: Proceedings CONCUR'93, 4th International Conference on Concurrency Theory. *Springer-Verlag Lecture Notes in Computer Science* **715** 66–81.

van Glabbeek, R. J. (2001) The linear time–branching time spectrum I; the semantics of concrete, sequential processes. In: Bergstra, J. A., Ponse, A. and Smolka, S. A. (eds.) *Handbook of Process Algebra*, Elsevier chapter 1, 3–99.

Welch, P., Brown, N., Morres, J., Chalmers K. and Sputh B. (2007) Integrating and extending JCSP. In: *Communicating Process Architectures* 48–76.