Theoretical Computer Science 36 (1985) 21-25
North-Holland

DETERMINACY $\longrightarrow$ (OBSERVATION EQUIVALENCE = TRACE EQUIVALENCE)

Joost ENGELFRIET*

Department of Computer Science, Twente University of Technology, 7500 AE Enschede, The Netherlands

Communicated by R. Milner
Received January 1984
Revised May 1984

Abstract. If an experiment s is conducted on a parallel process p, then, in general, different processes may result from the experiment, due to the nondeterministic behaviour of p (in the notation of Milner (1980): $p \stackrel{s}{\Rightarrow} p'$ for different p'). Process p is called determinate if the resulting processes are all equivalent (i.e., if $p \stackrel{s}{\Rightarrow} p'$ and $p \stackrel{s}{\Rightarrow} p''$, then p' and p'' are equivalent). This means that, although p behaves nondeterministically, this cannot be detected by an observer of p. Let $\simeq$ denote observation equivalence, used in CCS (Milner, 1980), let $\simeq_f$ denote (the much weaker) failure equivalence, used for CSP (Hoare et al., 1981; Brookes, 1983), and let $\simeq_t$ denote (the still weaker) trace equivalence. We show that the three corresponding notions of determinacy are the same, and that for determinate processes $\simeq$, $\simeq_f$, and $\simeq_t$ are the same. Determinacy is preserved under $\simeq$ and $\simeq_f$, but not under $\simeq_t$.

1. Preliminaries

We use the general model of parallel computation suggested in [4, Section 3.3] (see also [5] and [2]). We consider a tuple $(P, \Sigma, \{\stackrel{\mu}{\rightarrow}\})$, where P is the set of *programs* (or agents, states, etc.), Σ is the finite alphabet of *actions*, and for every $\mu \in \Sigma \cup \{\tau\}$, $\stackrel{\mu}{\rightarrow}$ is a binary relation on P; τ is a special symbol called the *unobservable action*, $\tau \notin \Sigma$. For every $w \in (\Sigma \cup \{\tau\})^*$ the relation $\stackrel{w}{\rightarrow}$ on P is defined in the usual way. For $s \in \Sigma^*$, the relation $\stackrel{s}{\Rightarrow}$ on P is defined by: $p \stackrel{s}{\Rightarrow} p'$ iff there exists $w \in (\Sigma \cup \{\tau\})^*$ such that $p \stackrel{w}{\rightarrow} p'$ and s is obtained from w by erasing all occurrences of τ. Intuitively, $p \stackrel{s}{\Rightarrow} p'$ means that p transforms into p' by experiment s.

For every program $p \in P$ we define the following sets, see [1]:

$$\text{trace}(p) = \{s \in \Sigma^* \mid p \stackrel{s}{\Rightarrow} p' \text{ for some } p' \in P\},$$

$$\text{init}(p) = \text{trace}(p) \cap \Sigma = \{a \in \Sigma \mid p \stackrel{a}{\Rightarrow} p' \text{ for some } p'\},$$

$$\text{fail}(p) = \{(s, X) \mid s \in \Sigma^*, X \subseteq \Sigma, \text{ and, for some } p' \in P:$$

$$p \stackrel{s}{\Rightarrow} p' \text{ and } X \cap \text{init}(p') = \emptyset\}.$$

* Present affiliation: Department of Computer Science, University of Leiden, P.O. Box 9512, 2300 RA Leiden, The Netherlands.

We consider the following equivalence relations on P (see [4, 5, 1, 2]): (a)-(e).

(a) *Observation equivalence* ($\simeq$). If R is a relation on P, then $F(R)$ is the relation on P defined by: $p\,F(R)\,q$ iff

(i) if $p \stackrel{s}{\Rightarrow} p'$, then $\exists q' : q \stackrel{s}{\Rightarrow} q'$ and $p'\,R\,q'$

(ii) if $q \stackrel{s}{\Rightarrow} q'$, then $\exists p' : p \stackrel{s}{\Rightarrow} p'$ and $p'\,R\,q'$.

R is a *bisimulation* if $R \subseteq F(R)$. Since F is monotonic (with respect to $\subseteq$), it follows from [6] that there is a maximal bisimulation $\simeq$ with $\simeq = \bigcup \{R \mid R \text{ is a bisimulation}\}$. (Note that the union of bisimulations is again a bisimulation.) It is easy to see that $\simeq$ is an equivalence relation on P.

(b) *Observation k-equivalence* ($\simeq_k$). For every $k \geqslant 0$, $\simeq_k$ is the equivalence relation on P, defined by

$$\begin{cases} \simeq_0 = P \times P, \\ \simeq_{k+1} = F(\simeq_k). \end{cases}$$

Moreover, $\simeq_\omega = \bigcap \{\simeq_k \mid k \geqslant 0\}$ is observation ω-equivalence. In [4], $\simeq_\omega$ is denoted $\simeq$ and called observation equivalence, but in [5] the bisimulation definition is used. Note that $\simeq \subseteq \simeq_\omega \subseteq \simeq_{k+1} \subseteq \simeq_k$.

(c) *Trace equivalence* ($\simeq_t$). For $p, q \in P$, $p \simeq_t q$ iff $\text{trace}(p) = \text{trace}(q)$. This means that the same experiments can be conducted on p and q. Note that $\simeq_t$ and $\simeq_1$ are the same.

(d) *Initial equivalence* ($\simeq_{\text{init}}$). For $p, q \in P$, $p \simeq_{\text{init}} q$ iff $\text{init}(p) = \text{init}(q)$. This means that p and q have the same initial capabilities. Note that $\simeq_t \subseteq \simeq_{\text{init}}$.

(e) *Failure equivalence* ($\simeq_f$). For programs p and q, $p \simeq_f q$ iff $\text{fail}(p) = \text{fail}(q)$. Also, $p \subseteq_f q$ iff $\text{fail}(q) \subseteq \text{fail}(p)$; i.e., p is 'less deterministic' than q [3, 1].

It is easy to see (cf. [1]) that $\simeq_2 \subseteq \simeq_f \subseteq \simeq_1$ ("f is $1\frac{1}{2}$"), and so $\simeq \subseteq \simeq_\omega \subseteq \cdots \subseteq \simeq_{k+1} \subseteq \simeq_k \subseteq \cdots \subseteq \simeq_2 \subseteq \simeq_f \subseteq \simeq_1 = \simeq_t \subseteq \simeq_{\text{init}}$. In fact, the following lemma is helpful in understanding the place of $\simeq_f$ in the sequence.

Lemma 1. *For $p, q \in P$, $\text{fail}(q) \subseteq \text{fail}(p)$ iff whenever $q \stackrel{s}{\Rightarrow} q'$, then $\exists p' : p \stackrel{s}{\Rightarrow} p'$ and $\text{init}(p') \subseteq \text{init}(q')$.*

Proof. ($\Rightarrow$) Consider (s, X) with $X = \Sigma - \text{init}(q')$.

($\Leftarrow$) Obvious. □

From this, $\simeq_f \subseteq \simeq_1$ is immediate, and $\simeq_2 \subseteq \simeq_f$ follows from the fact that, since $\simeq_1 \subseteq \simeq_{\text{init}}$, $\simeq_2 = F(\simeq_1) \subseteq F(\simeq_{\text{init}}) \subseteq \simeq_f$.

2. Results

We consider determinacy.

Definition. Let $\equiv$ be an equivalence relation on P. Program $p \in P$ is *$\equiv$-determinate* iff whenever $p \stackrel{s}{\Rightarrow} p'$ and $p \stackrel{s}{\Rightarrow} p''$, then $p' \equiv p''$.

Thus we have observation determinacy (with $\equiv = \simeq$), failure determinacy (with $\simeq_f$), trace determinacy (with $\simeq_t$), initial determinacy (with $\simeq_{init}$), etc. Perhaps this terminology is not completely appropriate. In fact, initial determinacy should perhaps be called failure determinacy, because it means that if $p \stackrel{s}{\Rightarrow} p'$ and $p \stackrel{s}{\Rightarrow} p''$, then p' and p'' have the same refusal sets (X is a refusal set of p' iff $X \cap \text{init}(p') = \emptyset$; clearly, p' and p'' have the same refusal sets iff $\text{init}(p') = \text{init}(p'')$).

Observation determinacy (in the $\simeq_\omega$-sense) is discussed in [4, Chapter 10] but not defined as such; it is a consequence of the OCD property, defined on [4, p. 155], as can be seen from [4, p. 156, the first three lines together with Theorem 10.14]. Also on this page, Milner mentions that every program in the determinate subcalculus DCCS has the OCD property (and hence is $\simeq_\omega$-determinate). Most of the examples in [4] can be expressed in DCCS.

Note that if $\equiv$ is equality, then determinacy is determinism, i.e., a program p is $=$-determinate iff p is deterministic. Thus every deterministic program is observation determinate, but not vice versa.

Our *results* are the following three.

(1) Observation determinacy and initial determinacy are the same (and hence $\equiv$-determinacy is the same for equivalences (a)-(e)). Therefore, we just call this: determinacy.

(2) For determinate programs, observation equivalence and trace equivalence are the same (and hence all mentioned equivalences, except initial equivalence).

(3) Determinacy is preserved under failure equivalence (and hence under all mentioned equivalences, except trace and initial equivalence). Even, if p is determinate and $p \subseteq_f q$, then q is determinate and $p \simeq q$.

In what follows we prove (1), (2), and (3). We note that fact (2) implies that all the logical equivalences discussed in [2] are the same as $\simeq$ and $\simeq_t$ for determinate programs (because, as shown in [2], they lie between $\simeq_\omega$ and $\simeq_f$).

The proof of (1) and (3) is based on the following lemma.

Lemma 2. *For all p and q in P,* (A) *implies* (B), *and* (B) *implies* (C), *where*

(A) *if $p \stackrel{s}{\Rightarrow} p'$ and $q \stackrel{s}{\Rightarrow} q'$, then $p' \simeq_{init} q'$,*

(B) *if $p \stackrel{s}{\Rightarrow} p'$ and $q \stackrel{s}{\Rightarrow} q'$, then $p' \simeq q'$,*

(C) *$p \simeq q$, and both p and q are $\simeq$-determinate.*

Proof. [(A) implies (B)]. Let $p, q \in P$ such that (A) holds. Define the relation R on P by: $p' R q'$ iff $\exists s \in \Sigma^* : p \stackrel{s}{\Rightarrow} p'$ and $q \stackrel{s}{\Rightarrow} q'$. Thus (A) means that $R \subseteq \simeq_{init}$, and we have to show (B), i.e., $R \subseteq \simeq$. We first show that $R \subseteq \simeq_t$, i.e., if $p' R q'$, then $\text{trace}(p') = \text{trace}(q')$. By symmetry, it suffices to show that: if $r \in \text{trace}(p')$, then $r \in \text{trace}(q')$. We argue by induction on the length of r. If r is empty, this is obvious (the empty string is in $\text{trace}(p')$ for every p'). Now consider ra, with $a \in \Sigma$, and assume the implication for r. Assume that $ra \in \text{trace}(p')$. Then there exist p'' and p''' such that $p' \stackrel{r}{\Rightarrow} p'' \stackrel{a}{\Rightarrow} p'''$. Since $r \in \text{trace}(p')$, by induction also $r \in \text{trace}(q')$, and so $q' \stackrel{r}{\Rightarrow} q''$ for some q''. Then clearly $p'' R q''$ (if $p \stackrel{s}{\Rightarrow} p'$ and $q \stackrel{s}{\Rightarrow} q'$, then $p \stackrel{sr}{\Rightarrow} p''$ and

$q \stackrel{sr}{\Rightarrow} q''$). Hence, by (A), $\text{init}(p'') = \text{init}(q'')$. Since $a \in \text{init}(p'')$, also $a \in \text{init}(q'')$, and so $q'' \stackrel{a}{\Rightarrow} q'''$. Hence $q' \stackrel{r}{\Rightarrow} q'' \stackrel{a}{\Rightarrow} q'''$, which shows that $ra \in \text{trace}(q')$. This proves that $R \subseteq \simeq_t$.

We now prove that R is a bisimulation. By symmetry, we only have to show that if $p' \, R \, q'$, then: if $p' \stackrel{r}{\Rightarrow} p''$, then $\exists q'': q' \stackrel{r}{\Rightarrow} q''$ and $p'' \, R \, q''$. Let $p \stackrel{s}{\Rightarrow} p'$ and $q \stackrel{s}{\Rightarrow} q'$. Since $R \subseteq \simeq_t$, $p' \simeq_t q'$. Now, if $p' \stackrel{r}{\Rightarrow} p''$, then $\exists q'': q' \stackrel{r}{\Rightarrow} q''$. Moreover, $p \stackrel{sr}{\Rightarrow} p''$ and $q \stackrel{sr}{\Rightarrow} q''$, and so $p'' \, R \, q''$. This shows that R is a bisimulation, and hence $R \subseteq \simeq$.

[(B) implies (C)]. Taking s empty in (B) shows that $p \simeq q$. Assume now that $p \stackrel{s}{\Rightarrow} p'$ and $p \stackrel{s}{\Rightarrow} p''$. Since $p \simeq_t q$, there exists a q' such that $q \stackrel{s}{\Rightarrow} q'$. By (B), $p' \simeq q'$ and $p'' \simeq q'$. Hence $p' \simeq p''$, and p is $\simeq$-determinate. By symmetry, so is q. □

It is easy to see that actually (A), (B), and (C) are equivalent statements.

Proof of (1). Clearly, if p is observation determinate, then p is initial determinate. The reverse implication follows by taking $q = p$ in Lemma 2. □

Proof of (2). Let p and q be determinate programs, and assume $p \simeq_t q$. We have to show that $p \simeq q$. Define R as in the proof of Lemma 2: $p' \, R \, q'$ iff $\exists s: p \stackrel{s}{\Rightarrow} p'$ and $q \stackrel{s}{\Rightarrow} q'$. We show again that R is a bisimulation (and then $R \subseteq \simeq$; note that $p \, R \, q$). By symmetry it suffices to show that if $p' \, R \, q'$ and $p' \stackrel{r}{\Rightarrow} p''$, then $\exists q'': q' \stackrel{r}{\Rightarrow} q''$ and $p'' \, R \, q''$. Assume $p \stackrel{s}{\Rightarrow} p'$, $q \stackrel{s}{\Rightarrow} q'$, and $p' \stackrel{r}{\Rightarrow} p''$. Then $p \stackrel{sr}{\Rightarrow} p''$, and so, since $p \simeq_t q$, $\exists q_1', q_1''$ such that $q \stackrel{s}{\Rightarrow} q_1' \stackrel{r}{\Rightarrow} q_1''$. Then $q' \simeq_t q_1'$ because q is determinate. Hence $\exists q'': q' \stackrel{r}{\Rightarrow} q''$, and clearly $p'' \, R \, q''$ (by sr). □

Proof of (3). Let p be determinate and assume that $p \subseteq_f q$. This means that $\text{fail}(q) \subseteq \text{fail}(p)$, i.e., by Lemma 1, if $q \stackrel{s}{\Rightarrow} q'$, then $\exists p': p \stackrel{s}{\Rightarrow} p'$ and $\text{init}(p') \subseteq \text{init}(q')$. We first show that, in fact, for that p', $\text{init}(p') = \text{init}(q')$. Let $a \in \text{init}(q')$, i.e., $q' \stackrel{a}{\Rightarrow} q''$. Then $q \stackrel{sa}{\Rightarrow} q''$. Hence, since $\text{fail}(q) \subseteq \text{fail}(p)$, $\exists p'': p \stackrel{sa}{\Rightarrow} p''$. So $\exists p''': p \stackrel{s}{\Rightarrow} p''' \stackrel{a}{\Rightarrow} p''$. Since p is determinate, $\text{init}(p''') = \text{init}(p')$, and so $a \in \text{init}(p')$.

Now we want to show (in order to use Lemma 2) that if $p \stackrel{s}{\Rightarrow} p'$ and $q \stackrel{s}{\Rightarrow} q'$, then $\text{init}(p') = \text{init}(q')$. Let $p \stackrel{s}{\Rightarrow} p'$ and $q \stackrel{s}{\Rightarrow} q'$. By the above (applied to $q \stackrel{s}{\Rightarrow} q'$), $\exists p'': p \stackrel{s}{\Rightarrow} p''$ and $\text{init}(p'') = \text{init}(q')$. Since p is determinate, $\text{init}(p') = \text{init}(p'') = \text{init}(q')$.

Thus (A) of Lemma 2 holds, and therefore also (C). Hence $p \simeq q$, and q is determinate. □

Note that determinacy is not preserved under trace equivalence. E.g.,

$$a.(b.\text{NIL} + c.\text{NIL}) \simeq_t a.b.\text{NIL} + a.c.\text{NIL},$$

and the first is determinate, but the second is not (cf. [4] for notation).

We finally note that we did not make use of the finiteness of Σ, so all our considerations also hold for infinite Σ. However, in [3], $\text{fail}(p)$ contains only (s, X) with finite X, even if Σ is infinite. For the resulting alternative definition of $\simeq_f$ and

$\subseteq_f$ our results still hold. In fact, then the only invalid statement is Lemma 1, but it is still true for determinate p, which is the case needed in the proof of (3). The details are left to the reader.

3. Conclusion

To prove $p \simeq q$ for programs p and q, it suffices to show $p \simeq_t q$ if it is known that both p and q are determinate, and it suffices to show $p \simeq_f q$ if it is only known that one of p and q is determinate (and it suffices to show (A) of Lemma 2 if it is not known that p and q are determinate).

Acknowledgment

I would like to thank Lex Augusteijn, Ed Brinksma, Maarten Fokkinga, Gerrit van der Hoeven, and Leo Verbeek for many interesting discussions.

References

[1] S.D. Brookes, On the relationship of CCS and CSP, in: *10th ICALP*, Lecture Notes in Computer Science **154** (Springer, Berlin, 1983) 83-96.

[2] S.D. Brookes and W.C. Rounds, Behaviourial equivalence of relations induced by programming logics, in: *10th ICALP*, Lecture Notes in Computer Science **154** (Springer, Berlin, 1983) 97-108.

[3] C.A.R. Hoare, S.D. Brookes and A.W. Roscoe, A theory of communicating sequential processes, Technical Monograph PRG-16, Oxford University, 1981; also: *J. ACM* **31** (1984) 560-599.

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

[5] R. Milner, Calculi for synchrony and asynchrony, *Theoret. Comput. Sci.* **25** (3) (1983) 267-310.

[6] A. Tarski, A lattice-theoretical fixpoint theorem and its applications, *Pacific J. Math.* **5** (1955) 285-309.