When is Partial Trace Equivalence Adequate?

Bard Bloom

Department of Computer Science, Cornell University, Ithaca, NY, USA

Keywords: Process algebra; Trace semantics; Structured operational semantics; Determinism

Abstract. Two processes are partial trace equivalent iff they can perform the same sequences of actions in isolation. Partial trace equivalence is perhaps the simplest possible notion of process equivalence. In general, it is too simple: it is not usually an adequate semantics. We investigate the circumstances under which it is adequate, which are surprisingly rich. We give two substantial classes of languages for which partial traces are adequate. In one class, partial trace equivalence suffices for total correctness, and operations such as true sequencing are possible; but all processes are determinate and silent moves are not possible. The other class - which includes many standard process calculi, such as CCS and CSP - admits indeterminacy and silent moves, but partial traces only suffice for partial correctness and true sequencing is not definable.

1. Introduction

The last decade has seen an explosion of research into simple languages for concurrency. This interest is roughly analogous to the interest in the simply-typed \( \lambda \)-calculus: both process algebras and simply-typed \( \lambda \)-calculus model elemental aspects of concurrency and higher-order programming respectively, allowing elucidation of fundamental issues without unnecessary complexity. One of the central questions of this research has been, “when are two processes equivalent?” [AbV90, BeK86, BIM90, BIM90, BoC89, BHR84, dNH84, G1G89, vGW89, Hen83, Mil83, PIP90, Wal90, KLP90, Gla90]. Answers to this question inform

1 Supported by NSF grant (CCR-9003441)

Correspondence and offprint requests to: Bard Bloom, Department of Computer Science, Cornell University, Ithaca, NY 14853, USA. Email: bard@cs.cornell.edu
most aspects of the theory and practice of concurrency, from language design to optimizing compilers.

Many notions of process equivalence arise as notions of congruence. Choose a set of aspects of process behaviour which are to be considered important; this is the set of observations. Describe the set of ways in which processes will be used; this is the set of contexts. Two processes are congruent (with respect to the given observations and contexts) if the same observations obtain in all contexts. If the observations and contexts are the right ones for the situation, the congruence is precisely the right notion of process equivalence.

Many equivalences are chosen by taking a simple programming language and notion of observation, and characterizing the appropriate notions of congruence, generally characterizing it in a somewhat language-independent way. This is the methodology of [BIM90, BHR84, dNH84, GrV89].

In this paper, we work in the other direction. We start with perhaps the simplest semantic model, the partial trace model, and describe a large class of languages for which this model is appropriate. This study is part of a larger research program linking semantic models and classes of programming languages.

The partial trace model (in one form or another) has been rediscovered by almost every researcher working on concurrency in any of its aspects. It is simple and appealing. A process is a construct capable of performing actions; two processes are considered identical iff they can perform the same finite sequences of actions.

It is well-known that partial traces are not adequate for most process algebras; we remind the reader of the counterexamples in section 1.2. However, the partial trace model is one of the most straightforward and elementary models of concurrency, easily explained and easily used. It would occasionally be useful to be able to design languages for which partial traces are adequate.

**Definition 1.1.** A language \( \mathcal{L} \) is partial trace respecting if partial traces give an adequate semantics; that is, whenever \( P \) and \( Q \) have the same partial traces, so do \( \mathcal{C}[P] \) and \( \mathcal{C}[Q] \) for all \( \mathcal{C}[-] \)-contexts \( C[-] \).

In this paper, we give sufficient conditions on the adequacy of partial traces: two incomparable classes of languages in which partial traces give adequate semantics. The first class, the partial trace determinate languages, places substantial restrictions on both processes and operations, resulting in a class of languages for which partial traces give all the behaviour of processes, and may be used to specify total correctness. The second class, the straight tyft languages, is more general and thus weaker; partial traces are good enough to specify partial correctness, but not total correctness, in straight tyft languages.

To investigate this area, we need some way to describe classes of languages. Most calculi for concurrency discussed above are presented as Structured Operational Semantics (SOS) [Plo81, Blo89, dSi85, GrV89], in which the behaviour of composite process terms is defined in terms of the behaviour of their subterms. We describe classes of languages in terms of their SOS definitions. This approach, of classes of SOS languages, has several advantages:

1. It is easy to test the conditions, by inspection of the rules of the language.
2. It is easy to define a partial trace respecting language, simply by working inside one of the classes provided.
3. It is frequently desirable to extend languages. In general, extending a language requires changes in the notion of process equivalence the language uses.