Failures semantics based on interval semiwords is a congruence for refinement

Walter Vogler*
Institut für Informatik, Technische Universität, Arcisstrasse 21, W-8000 München 2, Federal Republic of Germany

Received September 22, 1989 / Accepted September 19, 1990

Abstract. In this paper concurrent systems are modelled with safe Petri nets. The coarsest equivalence contained in failures equivalence is determined that is a congruence with respect to refinement. It is shown that in this context partial orders are necessary for the description of system runs: what is needed here are so-called interval orders.

Key words: Petri nets - Action refinement - Partial order semantics - Interval orders - Failures semantics - Divergence

1 Introduction

Concurrent systems can be modelled by process algebras like CCS [23], TCSP [8], or ACP [4], or by Petri nets [7]. Process algebras have the advantage of being modular by definition and having well-developed equivalence notions like bisimulation, failures or testing equivalence, algebraic laws and complete proof systems. Petri nets on the other hand offer a clear description of concurrency, a long tradition in the study of 'true' concurrency, and a graphical representation. Also a lot of effort has been put into the study of modular construction of Petri nets, especially for transition refinement.

Recently there has been some new interest in the question of refinement, especially since the refinement issue might shed some light on the old dispute of interleaving vs. true concurrency. In [28] and [9] it is argued that partial order semantics is useful when dealing with refinements. This is the topic of this paper.

In this paper we study refinements of actions using safe Petri nets as a model for concurrent systems. One reason for this choice is that it is fairly easy to define a satisfactory refinement operator on nets as a substitution. Thus, if we give a semantics to nets we immediately have a semantics of refined nets, since they are simply nets. (The same holds for event structures in the sense of [26], which can be seen as a restricted class of nets.) Also we can make use of existing studies regarding the refinement of Petri nets. On the other hand, process algebras usually do not offer a refinement operator. Simple syntactic substitution is not satisfactory here, at least in the presence of synchronization, see [2], [25], and [1] for first studies of process algebras without synchronization. Still, in the light of recent papers connecting Petri nets and process algebras ([16, 10, 17, 32]) it is hoped that our results also have some impact on the study of refinement in process algebras.

So far in the Petri net literature mostly transition refinements were considered and it was studied in which way they preserve behaviour ([24, 31, 33, 34]). The view on refinements in this paper is different. Here we want to replace an action \( a \) for example by a sequence \( a_1 a_2 \) of actions. With such a refinement we cannot expect that the refined net has the same behaviour as the original net, since totally different actions are possible in the two nets. Instead we expect that if two nets with equivalent behaviour are refined in the same way then the resulting nets have equivalent behaviour again. In other words, we should have a behaviour notion which is a congruence with respect to action refinement.

* This work was partially supported by the ESPRIT Basic Research Action No. 3148 DEMON (Design Methods Based on Nets)
The starting point of our considerations is the failures semantics developed for TCSP [8]. It is purely an inter-leaving semantics and has been shown to be useful, if one constructs nets with a parallel composition operator with synchronization and is mainly interested in the deadlocking behaviour of systems [35]. We study how this failures semantics has to be modified to become a congruence for refinements, i.e. we determine the coarsest equivalence contained in failures equivalence that is a congruence with respect to action refinement. It turns out that we must not describe system runs by sequences of actions, but have to use some partial order semantics.

In the literature there exists a variety of partial order semantics for nets. We find processes, see e.g. [18] or [5], and traces, see e.g. [22], partial words of nets are studied in [19] – also called partial order computations [29] or pomsets [28] –, and we also find a slight restriction of partial words, namely semiwords [30], which for safe nets coincide with partial words. Here we work with safe nets, and we will introduce a special class of semiwords, so called interval semiwords, specified by the requirement that the partial order involved is an interval order. (A partial order (A, <) is an interval order if for each x ≤ A we find a closed interval in R such that x < y if and only if the interval of x is totally to the left of the interval of y.) The idea is that when observing a system run we see actions starting and finishing, i.e. the execution of an action corresponds to some interval, and we order these action occurrences such that one is less than the other if it finishes before the other starts. Overlapping intervals correspond to concurrent actions. Thus an observer can see that some action is concurrent to a sequence of some other actions, i.e. he or she can see more than a sequence of steps, where a step is a set or multiset of simultaneous actions. But, as we will see, an observer cannot directly observe causality, as it is modelled e.g. by Petri net processes.

The congruence we are looking for can be defined similarly to the (sequential) failures semantics, but based upon interval semiwords instead of sequences. Thus we can prove what has been suspected for long: partial order semantics is needed when dealing with refinements. But we also show that we do not need the full strength of the usual partial order semantics of Petri nets. Interval semiwords are semiwords, but in general semiwords might 'contain more information on concurrency' than interval semiwords. On the other hand, interval semiwords are more distinctive than step sequences, i.e. step sequences are interval semiwords, but in general interval semiwords might 'contain more information on concurrency' than step sequences. One could say that the issue of this paper is to determine the right 'amount of information on concurrency' needed to deal with refinement. We also shortly discuss semiwords and the so called event structures of processes.

Recently, a number of related results have been obtained independently. The idea behind interval semiwords is closely related to the one underlying ST-bisimilarity, which was already defined in [16]. Very recently it has been shown in [13] that this equivalence is preserved by conflict-free refinements for event structures without internal actions. The ST-traces defined in [13] are a different representation of interval semiwords. Interval semiwords are also used in [25] for a semantics of a process algebra (which so far lacks communication); in that paper also a coarsest congruence result with respect to refinement is shown. For refinements of Petri nets congruence results based on bisimulation are given in [6] and [11]. Conflict-free refinements of event structures are studied in [14] and [36]; some congruence results are given, mostly based on bisimulation. In [14] also a congruence result is shown that corresponds to the result on event structures of processes we give below.

This paper is the only one so far which is concerned with failures semantics and action refinement. Especially, for the first time divergence is studied in this context.

In Sect. 2 some preliminaries are presented including two versions of the sequential failures semantics for nets we start from; one version takes divergence (i.e. infinite internal looping) into account, the other does not. In Sect. 3 the refinement we use here is defined. Sect. 4 deals with semiwords and interval semiwords. We introduce semiwords with termination set; these are based on the idea that a (part of a) system run is not only described by a partial order of transition firings, but also by recording when transition firings have terminated and which have started but have not terminated yet. The termination set is a somewhat technical, but important detail. Sect. 5 shows how to calculate semiwords of a refined net from semiwords of the original net and of the inserted nets. The results are formulated in quite a general way and allow to exhibit three simple congruences first; if we define a linear time semantics based on interval semiwords, or on semiwords in general, or on event structures of processes, then each of the resulting semantical equivalences is a congruence for refinement. In Sects. 6 and 7 the new failures semantics based on interval semiwords is presented and shown to induce the coarsest congruence with respect to refinement contained in the sequential failures equivalence. Sect. 6 treats failures without divergence, Sect. 7 failures with divergence. In the first case it is also possible to base failures equivalences on semiwords in general or on event structures of processes, such that these equivalences are congruences for refinement – although of course not the coarsest congruence we want to study here. But if divergence is to be considered our proofs do not carry over, since they depend on the special properties of interval orders. It is not clear how congruences based on semiwords or event structures of processes can be constructed in this case.

An extended abstract of a preliminary version of this paper has appeared as [37].

2 Preliminaries

Let \( \Sigma \) be an infinite set of actions. A labelled Petri net (short: a net) is a tuple \( N = (S_N, T_N, W_N, M_N, I_N) \) consisting of the (not necessarily finite) disjoint sets \( S_N \) of places and \( T_N \) of transitions, the weight function \( W_N : S_N \times T_N \cup T_N \times S_N \rightarrow N_0 \), the initial marking \( M_N : S_N \rightarrow N_0 \)