INFORMATION AND COMPUTATION **127**, 41–61 (1996)
ARTICLE NO. 0048

WALTER VOGLER*

Institut für Mathematik, Universität Augsburg, D-86135 Augsburg, Germany

Splitting is a simple form of action refinement that may be used to express the duration of actions. In particular, $split_n$ subdivides each action into n phases. Petri nets N and N' are $split_n$-language equivalent if $split_n(N)$ and $split_n(N')$ are language equivalent. It is known that these equivalences get finer and finer with increasing n. This paper characterizes the limit of this sequence by a newly defined partial order semantics. This semantics is obtained from the interval-semiword semantics, which is fully abstract for action refinement and language equivalence, by closing it under a special swap operation. The new swap equivalence lies strictly between interval-semiword and step-sequence equivalence. © 1996 Academic Press, Inc.

1. INTRODUCTION

Many models of concurrent systems assume that the actions which are performed by the system are instantaneous. The interleaving approach is based on this assumption: in this approach, which is traditionally followed in process algebra, the concurrent execution of two actions is regarded as being equal to performing them in any order. Hence, the behaviour of a system can be described by its language, which is the set of action sequences that the system can perform. This would not be adequate if actions had durations: then, concurrent actions could overlap in time and this would be observable. But also in Petri net theory, where traditionally a "true concurrency" approach is preferred, it is usually assumed that a transition firing is instantaneous.

In real life, however, actions usually take time. It is often assumed that we can nevertheless work with instantaneous actions: the suggestion is to replace an action with duration by a sequence of two instantaneous actions, one denoting the start, the other the end of the original action. (The first paper to work this out is probably [Hen88], which was conceived much earlier than 1988.) Hence, instead of considering the language of a system N, we first apply the operation $split_2$ to N, which splits each action a into a sequence $a_1 a_2$, and then consider the language of $split_2(N)$.

If this were a sensible treatment of durational actions, then, for each $n > 2$, splitting each action into n phases should give us the same information as splitting into two phases. To formulate this expectation mathematically, let the operation $split_n$ replace each action a with a sequence $a_1 \cdots a_n$; let us call systems N and N' *$split_n$-language equivalent* or simply *$split_n$-equivalent*, if the split systems $split_n(N)$ and $split_n(N')$ have the same language. With these notions, the above expectation says: if systems are $split_2$-equivalent, then they are $split_n$-equivalent for all n. Although plausible at first sight, this conjecture has turned out to be wrong, at least in the case of autoconcurrency where an action may be performed concurrently with itself. For all n, $split_{n+1}$-equivalence is strictly finer than $split_n$-equivalence [GV95]. Thus, if we want to model the durations of actions by considering them as sequences of several phases, we should work with the limit of these increasingly finer equivalences. The purpose of this paper is to characterize this limit; we will use safe Petri nets as system models, and we will give a characterization in terms of a new partial order semantics of nets, called swap-interval-semiwords.

Splitting is a simple form of action refinement. The latter operation is meant to support the hierarchical design of systems; it has recently found considerable interest, see e.g., [AH93, AM96, BDK91, DG95, Dev92, Gla90, GG89b, JM93, NEL89, Ren93, Vog92]. Action refinement replaces each action with a more detailed subsystem. A semantics supports this operation if it is possible to determine the semantics of the refined system from the refinement and the *semantics* of the system, i.e., without knowing the system itself. In other words, the semantics should induce a congruence with respect to action refinement.

In this context, a specific class of partial orders is of interest, so-called interval orders. The elements of interval orders correspond to intervals of real numbers, i.e. these partial orders are natural candidates for the description of system runs where actions have durations; see [Vog95] for some results in this direction. Processes [GR83] and semiwords [Gra81, Sta81] are well-established partial order semantics of Petri nets; interval semiwords combine the ideas of semiwords and interval orders. Process, semiword,

* Work on this paper was partially supported by the DFG (Project Halbordnungstesten), the ESPRIT Basic Research Working Group 6067 CALIBAN (CAusal CalcuLI BAsed on Nets) and the Fakultät für Informatik, Technische Universität München. This is a revised and extended version, an abstract of which has appeared in the Proceedings of ICALP 95, Lecture Notes in Computer Science Vol. 944, Springer-Verlag, Berlin/New York, 1995.

0890-5401/96 $18.00

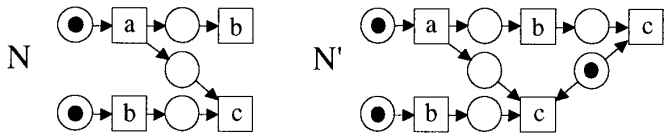


FIGURE 1

and interval-semiword equivalence are congruences for action refinement [Vog91, Vog92], but only interval-semiword equivalence has the following additional property: if nets N and N' are not interval-semiword equivalent, then there exists a refinement *ref* such that $ref(N)$ and $ref(N')$ are not language equivalent. Hence, if we take language-equivalence as a starting point, then interval semiwords make exactly those additional distinctions between nets which are necessary to get a congruence. Expressed in a technical way, interval-semiword equivalence is *fully abstract* with respect to language equivalence and action refinement, i.e., it is the coarsest congruence for action refinement that is finer than language equivalence. Analogously, the purpose of this paper is to determine a fully abstract semantics with respect to language equivalence and splitting. It is clear that interval semiwords induce a congruence for splitting that is finer than language equivalence, but as we will see, it is not the coarsest such congruence—as was already claimed without proof in [Lar88, Vog92]. At least, interval semiwords will be the starting point for the definition of the new fully abstract semantics.

The corresponding problem on the level of bisimulations (which do not simply consider the sequences of actions that are performed, but also take into account the choices that are possible during a system run) has been solved: as on the language level, $split_{n+1}$-bisimilarity is strictly finer than $split_n$-bisimilarity for all $n \in \mathbb{N}$ [GV95]. ST-bisimulation is a variant of bisimulation that corresponds to the interval idea. It is fully abstract with respect to bisimulation and general action refinement [Gla90, Vog93], but—quite surprisingly—it is also fully abstract with respect to bisimulation and splitting [GL95].

That the situation is different on the language level can be seen from the following example. If we want to describe the runs of the two nets in Fig. 1 by partial orders, then we get essentially that N can perform the (labelled) partial order p in Fig. 2, while N' can perform p and the partial order q. These partial orders are interval orders, hence N and N' have different interval semiwords.

The split nets $split_3(N)$ and $split_3(N')$ can perform the sequence $a_1 b_1 b_2 a_2 a_3 b_1 b_3 c_1$. From this sequence we can deduce that N and N' can perform p: in this sequence, we see an a starting, then a b. Since this first b starts before the end of a, it must be independent of a; it corresponds to the b in the lower left corner of p. Similarly, the second b is independent of the first one; but it comes after a. Now, b_3 is the end of some b; the essential point is, that it must in fact be the end of the first b, since only the first b has finished its second phase at this point. Hence, c comes after a and the first b, but it is independent of the second b.

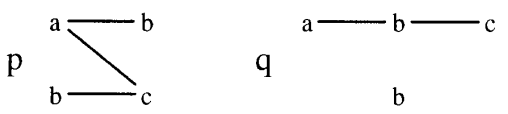


FIGURE 2

For q, we argue that there is no corresponding sequence in any $split_n(N')$. In such a sequence, we would see the lower b starting during the a-action. Hence, the upper b would start after the lower one, but it would end *before* the lower one, since c starts during the lower b. In such a situation, where one b overtakes the other, we cannot keep the two b's apart by keeping them in different phases—as we have done above for p. Hence, when some b ends, we cannot be sure which one it is. Consequently, we can only deduce that N' can perform q or p, something we already know. Thus, it seems convincing that N and N' cannot be differentiated by looking at the languages of any of their splittings, but they are distinguished by interval-semiword equivalence. Hence, this equivalence is different from the limit of the $split_n$-equivalences.

In the above example, the presence of p "hides" q. Here, and in similar cases, we can obtain q from p by applying a swap-operation to the two b-actions, which exchanges their successors in the partial order p. This example motivates the new semantics which characterizes the limit we are looking for: the swap-interval-semiwords of a net are obtained by closing the set of interval semiwords under the swap-operation.

Swap-interval-semiwords are a way to deal with durational actions in concurrent systems. But they can also be interesting in formal language theory. If we consider expressions built from variables for languages with the operations concatenation $\cdot$, choice $+$ (i.e., union), and shuffle $\|$, then substitution of languages for the variables can be seen as action refinement; validity of equations for such expressions can be checked using interval semiwords [Mey95]. Similarly, swap-interval-semiwords are useful to establish the validity of equations where the variables stand for words; e.g., from our results one can see that for all words X, Y, and Z we have

$$XY \parallel YZ = (XY \parallel YZ) + (XYZ \parallel Y).$$

This equation fails for languages.

In the rest of the paper, we proceed as follows. In Sections 2 and 3, we study interval orders and their representations by so-called interval sequences. These interval sequences are used in many of our proofs, but similar representations have also been studied in their own right; see, e.g., [JK93]. In Section 2, we characterize the set of representations of a

given interval order; furthermore, we look at the augmentations of an interval order and show how augmentation corresponds to an operation on interval sequences. These results are also interesting in their own right; an extension to infinite interval orders is discussed in a short appendix. Section 3 is concerned with the *swap* operation, which is defined on interval orders and their representations; a correspondence between the two versions of *swap* is shown. Section 4 introduces some Petri net notions, including interval-semiword semantics, and the split operations. In Section 5, the new swap-interval-semiword semantics is defined and the announced full abstractness result is given. Half of the proof of this result is deferred to Section 6; it is based on a more detailed version of interval sequences. In Section 5, we also show that the new semantics is closed under augmentation and prefixes. (Note that these relations are defined in a natural way for interval orders, but not for their representations.) With these results, we can really prove that the nets of Fig. 1 are indeed *split*$_n$-equivalent for all $n \in \mathbb{N}$. With this, we get that the new swap equivalence lies strictly between interval-semiword and step-sequence equivalence. We present two consequences of our main result in Section 7: we show that we can justify interval semiwords, when we additionally consider parallel composition with synchronization as in TCSP; i.e., interval-semiword equivalence is fully abstract for language equivalence, parallel composition and splitting. Furthermore, we sketch how the above mentioned equations can be checked. The paper closes with some concluding remarks in Section 8.

I am grateful to Rob van Glabbeek for discussing the above example and first ideas about swapping with me. I thank Roberto Gorrieri and two anonymous referees for their careful reading and constructive suggestions.

2. REPRESENTING AND AUGMENTING INTERVAL ORDERS

In this section, we look at interval orders and their representation by sequences; a basic reference for interval orders is [Fis85, Chap. 2]. Furthermore, we study interval orders that are augmentations of some given interval order and relate their representations to the representations of this given interval order.

We restrict attention to finite interval orders, since we are interested in finite system runs later on. We will describe system runs by labelled interval orders; but, since the concepts of this section are independent of the labels, we will introduce the labelling only in the next section.

A *partial order* $p = (E, <)$ consists of a finite set E (of *events*) and an irreflexive, transitive relation $<$ on E. We will consider isomorphic partial orders as equal. If $e < e'$, we call e a *predecessor* of e' and e' a *successor* of e. If $e < e'$ or $e = e'$, we write $e \leqslant e'$. Two events e and e' are *concurrent*, $e \; co \; e'$, if neither $e < e'$ nor $e' < e$ (but possibly $e = e'$). If co is transitive, we call p *co-transitive*. (In [Fis85] such a p is called a weak order.) In this case, co is an equivalence relation and concurrent events are ordered in the same way with respect to all other events; hence, $<$ induces an ordering of the co-equivalence classes, and we can think of p as a sequence of these classes.

$E' \subseteq E$ is *left-closed* if for all $e' < e \in E'$ we have $e' \in E'$. $(E', <')$ is a *prefix* of $(E, <)$ if E' is a left-closed subset of E and $<'$ equals $<$ restricted to $E' \times E'$. We can think of a prefix as an initial segment of the system run $(E, <)$.

$(E, <)$ is a (*proper*) *augmentation* of $(E, <')$, if $<'$ is a (proper) subset of $<$; if $(E, <')$ describes a system run, then $(E, <)$ describes essentially the same run, but includes some more ordering information about the events. If $<$ is *total*, i.e., for all events e and e' we have $e < e'$, $e = e'$, or $e' < e$, then $(E, <)$ is a *linearization* of $(E, <')$. In this case, we can view $(E, <)$ as a sequence; in this sequence, each event occurs exactly once. If X is a set of partial orders, we call $p \in X$ *least sequential* in X if p is not a proper augmentation of any $q \in X$.

$(E, <)$ is an *interval order*, if for all $a, b, c, d \in E$ we have: if $a < b$ and $c < d$, then $a \leqslant d$ or $c \leqslant b$. Thus, whenever we have the "parallel arrows" shown in Fig. 3, we must also have one of the "diagonals" indicated by the dotted lines. (We depict partial orders as Hasse diagrams, where a line connecting a and b from left to right indicates that $a < b$.) A (proper) augmentation p of q is called a (*proper*) *interval augmentation* of q if p is an interval order. The name "interval order" originates from the following result.

THEOREM 2.1. [Fis85]. *$(E, <)$ is an interval order if and only if there are closed intervals $[r_1(e), r_2(e)] \subseteq \mathbb{R}$, $e \in E$, such that all events e and e' satisfy the equivalence: $e < e'$ if and only if $r_2(e) < r_1(e')$.*

We can think of event e as starting at time $r_1(e)$ and lasting until $r_2(e)$. Event e is smaller than event e' if the interval of e lies completely before the interval of e'.

For a finite set E let $E^{\pm} = E^+ \dot{\cup} E^-$ be the disjoint union of two copies of E, where $E^+ = \{e^+ \mid e \in E\}$ and $E^- = \{e^- \mid e \in E\}$. We call e^+ the *start* of e and e^- the *end* of e. A sequence $w \in (E^{\pm})^*$ is called an *interval sequence* (*over* E) if for all $e \in E$:

— w contains e^+ once

— w contains e^- at most once, and if e^- occurs in w, it occurs after e^+.

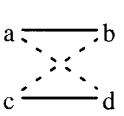


FIGURE 3

The interval sequence w is called *closed*, if e^- occurs in w for each $e \in E$.

Interval sequences w over E and w' over E' are *isomorphic* if there is some bijection $\beta: E \to E'$ such that w' can be obtained from w by replacing each e^+ by $\beta(e)^+$ and each e^- by $\beta(e)^-$. Since we are not so much interested in the concrete events, but in the structure of their starts and ends, we usually do not distinguish isomorphic interval sequences.

Given an interval sequence w, we can associate to each $e \in E$ an interval $[r_1(e), r_2(e)] \subseteq \mathbb{R}$ as follows: $r_1(e)$ is the position of e^+ in w, $r_2(e)$ is the position of e^- in w or (in case that e^- does not occur in w) some value greater than $|w|$, the length of w. In view of Theorem 2.1, we define:

DEFINITION 2.2. An interval sequence w over E *represents* (or is a *representation* of) an interval order $(E, <)$ if for all $e_1, e_2 \in E$ we have $e_1 < e_2$ if and only if e_1^- occurs before e_2^+ in w.

We can think of a representation as a sequential observation where we see events start and end; thus, we observe the events as intervals in time and from this we can conclude that some partial order of events occurred, namely the represented interval order. The ordering relation of this partial order is not so much causality based; but it is natural to assume that event e can cause event e', only if e ends before e' starts, i.e. if $e < e'$ in the interval order.

In [Vog92], it is shown that each interval order has some closed representation. Here, we want to find out about *all* representations. For this purpose, we define three relations $<^+$, $<^-$, $<^\pm$.

DEFINITION 2.3. For events e_1 and e_2 of an interval order $(E, <)$, we write $e_1 <^+ e_2$ if for some e_3 we have $e_1 \mathrel{co} e_3$ and $e_3 < e_2$. Analogously, $e_1 <^- e_2$ if for some e_3 we have $e_1 < e_3$ and $e_3 \mathrel{co} e_2$.

These relations are introduced for the following reason: if w represents $(E, <)$ and $e_3 < e_2$, then e_3^- occurs before e_2^+ in w; furthermore, $e_1 \mathrel{co} e_3$ implies that e_1^+ occurs before e_3^-, since otherwise e_3^- would not occur in w or we would have $e_3 < e_1$, which are both wrong. Hence, e_1^+ must occur before e_2^+ if $e_1 <^+ e_2$. Similarly, e_1^- must occur before e_2^- or e_2^- does not occur at all, if $e_1 <^- e_2$.

Since co is reflexive, we have that $e_1 < e_2$ implies $e_1 <^+ e_2$ and $e_1 <^- e_2$. We can characterize $<^+$ and $<^-$ as follows:

LEMMA 2.4. *Let $(E, <)$ be an interval order and $e_1, e_2 \in E$.*

(i) *$e_1 <^+ e_2$ if and only if the set of predecessors of e_1 is a proper subset of the set of predecessors of e_2.*

(ii) *$e_1 <^- e_2$ if and only if the set of successors of e_2 is a proper subset of the set of successors of e_1.*

Proof. (i) "$\Rightarrow$" Choose e_3 with $e_1 \mathrel{co} e_3 < e_2$ and $e_4 < e_1$. Since $(E, <)$ is an interval order, we have $e_3 \leqslant e_1$ or $e_4 \leqslant e_2$. If $e_3 \leqslant e_1$, we conclude that $e_3 = e_1$ and $e_4 < e_2$ by transitivity. If $e_4 \leqslant e_2$, we can exclude $e_4 = e_2$, since this would imply $e_3 < e_1$. Hence, the subset-property follows, and the subset is proper due to e_3.

"$\Leftarrow$" Choose e_3 such that $e_3 < e_2$, but not $e_3 < e_1$. If $e_1 < e_3$, we get $e_1 \mathrel{co} e_1 < e_2$; otherwise $e_1 \mathrel{co} e_3 < e_2$. Hence $e_1 <^+ e_2$.

(ii) Similar. ∎

DEFINITION 2.5. For an interval order $(E, <)$, we define $<^\pm$ on $E^\pm$ as follows:

— $e_1^+ <^\pm e_2^+$ if $e_1 <^+ e_2$

— $e_1^- <^\pm e_2^-$ if $e_1 <^- e_2$

— $e_1^- <^\pm e_2^+$ if $e_1 < e_2$

— $e_1^+ <^\pm e_2^-$ if $e_1 < e_2$ or $e_1 \mathrel{co} e_2$.

Similarly to the above, we get for a representation w of $(E, <)$:

— If $e_1^+ <^\pm e_2^+$, then e_1^+ occurs before e_2^+ in w; see above.

— If $e_1^- <^\pm e_2^-$, then e_1^- occurs before e_2^- in w, or e_2^- does not occur; see above.

— If $e_1^- <^\pm e_2^+$, then e_1^- occurs before e_2^+ in w, by definition of a representation.

— If $e_1^+ <^\pm e_2^-$, then e_2^- does not occur before e_1^+ by definition of a representation, hence e_1^+ occurs before e_2^- in w or e_2^- does not occur at all.

As a consequence, every closed representation of $(E, <)$ is an augmentation, hence a linearization of $(E^\pm, <^\pm)$.

We have the following result [Fis85]. (Observe that $<^+$ corresponds to $\prec^-$ in [Fis85] and vice versa.)

LEMMA 2.6. *Let $(E, <)$ be an interval order.*

(i) *$<^+$, $<^-$, $<^\pm$ are co-transitive partial orders with $< \subseteq <^+$ and $< \subseteq <^-$.*

(ii) *If $e_1^{\varepsilon_1} \mathrel{co} {}^\pm e_2^{\varepsilon_2}$ (i.e., $e_1^{\varepsilon_1}$ and $e_2^{\varepsilon_2}$ are concurrent with respect to $<^\pm$), then $\varepsilon_1 = \varepsilon_2$ (i. e., both equal − or both equal +).*

(iii) *$e^+ <^\pm e^-$.*

(iv) *If $e_1^+ <^\pm e_2^+$, then there exists some e_3 with $e_1^+ <^\pm e_3^- <^\pm e_2^+$.*

(v) *If $e_1^- <^\pm e_2^-$, then there exists some e_3 with $e_1^- <^\pm e_3^+ <^\pm e_2^-$.*

This lemma shows that the $co^\pm$-equivalence classes consist of starts only or of ends only ((ii)) and that start- and end-classes occur alternatingly ((iv) and (v)) beginning with a class of starts and ending with a class of ends ((iii)). With this lemma, we can characterize the representations of an interval order as follows—where the difficult part is that *all* (closed) representations can be obtained as described.

THEOREM 2.7. *Let $p=(E, <)$ be an interval order.*

(i) *The closed representations of p are exactly the linearizations of $(E^{\pm}, <^{\pm})$. In particular, there exists a closed representation of p.*

(ii) *If w is a closed representation of p, then exactly all closed representations of p can be obtained from w by commuting starts and commuting ends. More precisely, define a relation on $(E^{\pm})^*$ as follows: $w_1 e_1^+ e_2^+ w_2 \equiv w_1 e_2^+ e_1^+ w_2$ and $w_1 e_1^- e_2^- w_2 \equiv w_1 e_2^- e_1^- w_2$ for all $e_1, e_2 \in E$, $w_1, w_2 \in (E^{\pm})^*$, let $\equiv^*$ be the reflexive–transitive closure of $\equiv$. Then the $\equiv^*$-equivalence class of w is the set of all closed representations of p.*

(iii) *If w is a closed representation of p, then exactly all representations of p can be obtained from w by commuting starts, commuting ends (as in (ii)) and deleting ends from the end of a sequence (i.e., transforming $w'e^-$ to w').*

Proof. (i) We have already seen above that each closed representation is an augmentation and hence a linearization of $(E^{\pm}, <^{\pm})$. On the other hand, such a linearization w is an interval sequence by Lemma 2.6(iii); if $e_1 < e_2$, then $e_1^- <^{\pm} e_2^+$, i.e., e_1^- occurs before e_2^+ in w; if not $e_1 < e_2$, then $e_2 < e_1$ or $e_2 \, co \, e_1$, hence $e_2^+ <^{\pm} e_1^-$, i.e., e_1^- does not occur before e_2^+ in w. This shows that w is a representation of p.

(ii) Part (i) shows that each closed representation is obtained by putting the elements of each $co^{\pm}$-equivalence class into some arbitrary sequence and concatenating these sequences in the order prescribed by $<^{\pm}$. Since each $co^{\pm}$-equivalence class consists of starts only or of ends only, by 2.6(ii), we see that we can get all closed representations by commuting starts and commuting ends. On the other hand, starts that do not belong to the same $co^{\pm}$-equivalence class are separated by an end in $<^{\pm}$, see 2.6(iv), hence also in any closed representation. Thus, commuting starts exchanges starts from the same $co^{\pm}$-equivalence class and gives another closed representation. The case of commuting ends is analogous.

(iii) In view of Part (ii), we only have to consider the difference between closed representations and arbitrary representations, where ends of events may be missing. We have seen above that all representations are augmentations of $(E^{\pm}, <^{\pm})$, except that in general some ends might be missing. If $e_1^- <^{\pm} e_2^+$, then $e_1 < e_2$ and e_1^- must occur before e_2^+ in each representation. Hence, the missing ends must all belong to the maximal $co^{\pm}$-equivalence class. Thus, for an arbitrary representation w we can choose a closed representation w' that coincides with w except for the missing ends. By commuting ends in w' we can move these missing ends of w to the end of w' and by deleting ends we obtain w.

On the other hand, none of the ends belonging to the maximal $co^{\pm}$-equivalence class is needed in a representation, as is easily seen. Deleting ends as described in (iii) deletes only elements of this class, hence transforms a representation of p to another one. ∎

This theorem shows that interval orders are more abstract than interval sequences. Several interval sequences may represent the same interval order; they differ in the ordering of starts or in the ordering of ends, but these differences may be irrelevant for some purposes and are abstracted away in the interval order.

Remark. Theorem 2.7(ii) shows that the set of closed representations of some given p is a Mazurkiewicz trace (see, e.g., [Die90]) over $E^{\pm}$, where every two starts and every two ends are independent. An extension of Theorem 2.7 to the case of infinite interval orders is discussed in the appendix.

In the second half of this section we will have a look at the augmentations of a given partial order p. Suppose we want to prove some claim for all augmentations of $p=(E, <)$; in order to allow for an inductive proof of such a claim, we must have that each augmentation can be reached by adding to $<$ one pair at a time. We will show that this is indeed the case. Observe that not any pair will do; if we add the pair (b, c) to $<$ as shown in Fig. 4, then we destroy transitivity.

THEOREM 2.8. *Let $p=(E, <)$ and $p'=(E, <')$ be partial orders such that p' is a proper augmentation of p. Then there exists a partial order $p''=(E, <\cup\{(e_1, e_2)\})$ such that p'' is a proper augmentation of p and p' is an augmentation of p''.*

Proof. Choose e_1 minimal w.r.t. $<$ in the non-empty set $\{e \in E \mid \exists e' \in E: e <' e' \text{ and not } e < e'\}$; choose e_2 maximal w.r.t. $<$ in the non-empty set $\{e \in E \mid e_1 <' e \text{ and not } e_1 < e\}$. Obviously, $<'' = < \cup \{(e_1, e_2)\}$ is irreflexive. To show transitivity of $<''$, we first consider some $e < e_1$. Since $e <' e_1 <' e_2$, we have $e <' e_2$; by choice of e_1 we must have $e < e_2$. Second, let $e_2 < e$. Since $e_1 <' e_2 <' e$, we have $e_1 <' e$; by choice of e_2 we must have $e_1 < e$. ∎

Next, we want to show a similar result for interval orders. Consider the interval order p on the left hand side of Fig. 5 and its augmentation p' on the right hand side, which is an interval order, too.

FIGURE 4

FIGURE 5

The construction in the above proof makes it possible to choose $e_1 = c$ and $e_2 = d$. Adding the pair (c, d) to p gives indeed a partial order, but not an interval order; hence, we have to be more careful.

THEOREM 2.9. *Let $p = (E, <)$ be an interval order and let $p' = (E, <')$ be a proper interval augmentation of p. Then there exists a proper interval augmentation $p'' = (E, < \bigcup \{(e_1, e_2)\})$ of p such that p' is an augmentation of p''.*

Proof. Choose e_1 minimal w.r.t. $<^-$ (!) in the non-empty set $\{e \in E \mid \exists e' \in E\colon e <' e' \text{ and not } e < e'\}$; choose e_2 maximal w.r.t. $<^+$ in the non-empty set $\{e \in E \mid e_1 <' e \text{ and not } e_1 < e\}$. By Lemma 2.6(i) and the proof of Theorem 2.8, we get a partial order p'' with this choice.

We have to check that p'' is an interval order and, since p is an interval order, it is sufficient to consider e_1, e_2 and some c, d with $c < d$. Since p' is an interval order, we have $e_1 \leqslant' d$ or $c \leqslant' e_2$.

(a) $e_1 \leqslant' d$. If $e_2 < c$, then $e_1 \leqslant d$ by transitivity of p'' and we are done—also, of course, if $c \leqslant e_2$. Hence, let us assume that $e_2 \mathbin{co} c$. This implies that $e_2 <^+ d$. Since $e_1 \leqslant' d$, the choice of e_2 implies that $e_1 \leqslant d$.

(b) $c \leqslant' e_2$. If $d < e_1$, then $c \leqslant e_2$ by transitivity of p'' and we are done—also, of course, if $e_1 \leqslant d$. Hence, let us assume that $e_1 \mathbin{co} d$. This implies that $c <^- e_1$, thus c has the same successors in p and p' by choice of e_1, and we conclude that $c \leqslant e_2$. ∎

To conclude this section, we give a result for the representations of the augmentations of a given interval order. We show a lemma first.

LEMMA 2.10. *Let $p = (E, <)$ and $p' = (E, <')$ be interval orders such that p' is a proper augmentation of p with $<' = < \bigcup \{(e_1, e_2)\}$. Then there exists a closed representation $w_1 e_2^+ e_1^- w_2$ of p and, for any such closed representation, $w_1 e_1^- e_2^+ w_2$ is a representation of p'.*

Proof. The last claim is obvious, once we have shown the existence of $w_1 e_2^+ e_1^- w_2$. In view of 2.7(i) and 2.6, such a representation exists unless there is some e_3 with $e_2^+ \leqslant^{\pm} e_3^+ \leqslant^{\pm} e_1^-$. In this case, there exists some e_4 with $e_2 \mathbin{co} e_4 < e_3$; furthermore, we have $e_3 < e_1$ or $e_3 \mathbin{co} e_1$. In this situation, we have $e_1 <' e_2$ and $e_4 <' e_3$ and, therefore, $e_1 \leqslant' e_3$ or $e_4 \leqslant' e_2$, since p' is an interval order.

(a) $e_1 \leqslant' e_3$. Since we have not $e_1 < e_3$, this implies $e_2 = e_3$ or $e_1 = e_3$. The former implies $e_2 \mathbin{co} e_4 < e_2$, a contradiction. The latter gives $e_4 < e_1 <' e_2$, hence $e_4 <' e_2$ and by definition of p' also $e_4 < e_2$, a contradiction to $e_2 \mathbin{co} e_4$.

(b) $e_4 \leqslant' e_2$. Since we have $e_2 \mathbin{co} e_4$, this implies $e_1 = e_4$ or $e_2 = e_4$. The former implies $e_1 < e_3$, a contradiction in view of $e_3 < e_1$ or $e_3 \mathbin{co} e_1$. The latter implies $e_1 <' e_2 < e_3$, hence $e_1 <' e_3$ and by definition of p' also $e_1 < e_3$, again a contradiction. ∎

THEOREM 2.11. *Let p be an interval order and w a closed representation of p. Then v is a closed representation of some interval augmentation p' of p if and only if v can be obtained from w by commuting starts, commuting ends as in* 2.7(ii) *and moving starts to the end, i.e., transforming $w_1 e_2^+ e_1^- w_2$ to $w_1 e_1^- e_2^+ w_2$, where $e_1 \neq e_2$. The same holds for arbitrary representations if we additionally allow to delete ends from the end as in* 2.7(iii).

Proof. "⇒" In view of Theorem 2.7, this follows by induction on the size of the order relation from 2.9 and 2.10.

"⇐" Obvious.

3. SWAPPING

In this section, we introduce and study an operation on interval orders and their representations that we need for our later results on *split*$_n$-refinements. From now on, we will consider *labelled* partial orders, in particular *labelled* interval orders $(E, <, l)$, which consist of a partial or interval order $(E, <)$ and a *labelling* $l\colon E \to \Sigma$ of the events with elements from some fixed set Σ of *actions*. The definitions and results of Section 2 carry over to the labelled case, where the *labelled interval sequence* (w, l) is a *representation* of $(E, <, l)$ if w is a representation of $(E, <)$. When we depict a labelled partial order, we replace the events by their labels as in Fig. 2.

As explained in Section 2, we can view unlabelled (or, analogously, labelled) interval orders as observations where the events (or actions) have durations. Fig. 6 shows possible intervals during which the actions of p and q of Fig. 2 might take place. (To make the intervals visible, they are not drawn on the same line, but above each other.)

When we observe the labelled interval order p, we see two overlapping b-actions. The lower b starts first (namely, before the end of a) and later overlaps with the upper b. Then we see that the first b is followed by c, while the second b is independent of c. In the setting of *split*$_n$-refinement, such

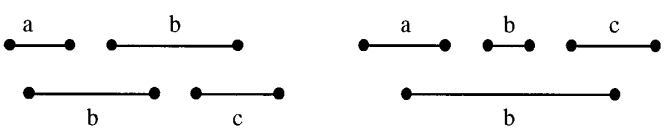


FIGURE 6

an overlap of one event with another might lead to confusing these events; in other words, we might get the impression that the upper b overtook the lower one and ended earlier such that in fact we observed q instead of p. We can obtain q from p by exchanging the successors of the two b-labelled events. This *swap-operation* is not symmetric: in p, the two b-events run in parallel, i.e., the one that starts first also ends first; we can transform p to q, where we have some overtaking, but not vice versa.

DEFINITION 3.1. Let $p=(E, <, l)$ be a labelled interval order such that for some $e_1, e_2 \in E$ we have $e_1 \, co \, e_2$, $e_1 <^+ e_2$, $e_1 <^- e_2$ and $l(e_1)=l(e_2)$. Let $q=(E, <', l)$ be defined by

$$e <' e' \text{ if } \begin{cases} e<e' & \text{and} \quad e_1 \neq e \neq e_2 \\ e_1<e' & \text{and} \quad e=e_2 \\ e_2<e' & \text{and} \quad e=e_1. \end{cases}$$

Then q is obtained from p by *swapping* (e_1 and e_2), denoted by $q \in swap(p)$. If some q' is obtained from p by applying this swap-operation arbitrarily often (including zero times), we write $q \in swap^*(p)$.

This operation is actually easier to understand on the level of interval sequences, where the starts and ends of events are treated explicitly. For this reason, we will use representations in our proofs, although we are ultimately interested in system runs described as partial orders—for which augmentation and prefix are intuitive notions. For example, we will rely on representations in our proof that the swap-operation on labelled interval orders is well-defined, i.e. that $q \in swap(p)$ is a labelled interval order again. But before defining the swap-operation on labelled interval sequences, we note the following.

PROPOSITION 3.2. *If $q \in swap(p)$, then $q \neq p$ (more precisely, p and q are not isomorphic).*

Proof. Let q be obtained by swapping e_1 and e_2 in $p=(E, <, l)$. Let e_1 have n predecessors in p, and let X be the set of events that have exactly n predecessors in p. Since $e_1 <^+ e_2$, we conclude from Lemma 2.4 that e_2 has more than n predecessors and $e_2 \notin X$. If some event e has different predecessors in p and q, then we must have $e_1 < e$ or $e_2 < e$; i.e., e must have more than n predecessors in p and in q. Hence, X is also the set of events that have exactly n predecessors in q and any isomorphism from p to q must map X to X.

Only e_1 and e_2 have different successors in p and in q. Since $e_1 <^- e_2$, we conclude from Lemma 2.4 that all events in X have the same number of successors in p and q except e_1 which has strictly less successors in q than in p. Hence, no isomorphism from p to q can map X to X, and we conclude that no isomorphism exists. ▮

DEFINITION 3.3. Let (w, l) be a labelled interval sequence such that for some e_1, e_2 we have that $l(e_1)=l(e_2)$, e_1^+ occurs before e_2^+ in w, e_2^+ before e_1^- and (provided e_2^- occurs at all) e_1^- before e_2^-. Let (v, l) be obtained from (w, l) by exchanging e_1^- and e_2^- in w. Then we say that (v, l) is obtained from (w, l) by *swapping* (e_1 and e_2), denoted by $(v, l) \in swap(w, l)$. If some (v', l) is obtained from (w, l) by applying this swap-operation arbitrarily often (including zero times), we write $(v', l) \in swap^*(w, l)$. ▮

Obviously, (v, l) as defined above is a labelled interval sequence. First we show the close interrelation between the swap-operation on interval orders and that on interval sequences. In particular, this will imply that $q \in swap(p)$ is indeed a labelled interval order.

LEMMA 3.4. *Let p be a labelled interval order. Then the following five statements are equivalent*:

(i) $q \in swap(p)$ *or* $p=q$.

(ii) *For each representation (w, l) of p there exists some representation (v, l) of q such that $(v, l) \in swap(w, l)$ or $(v, l)=(w, l)$.*

(iii) *For each representation (v, l) of q there exists some representation (w, l) of p such that $(v, l) \in swap(w, l)$ or $(v, l)=(w, l)$.*

(iv) *There exists some closed representation (w, l) of p and some representation (v, l) of q such that $(v, l) \in swap(w, l)$ or $(v, l)=(w, l)$.*

(v) *There exists some representation (w, l) of p and some representation (v, l) of q such that $(v, l) \in swap(w, l)$ or $(v, l)=(w, l)$.*

Proof. (i) $\Rightarrow$ (ii) Let (w, l) be a representation of $p=(E, <, l)$. If $p=q$, we can choose $(v, l)=(w, l)$. Hence, let $q=(E, <', l)$ be obtained from p by swapping e_1 and e_2. Since $e_1 <^+ e_2$, e_1^+ occurs before e_2^+ in w; since $e_1 \, co \, e_2$, e_2^+ occurs before e_1^- in w or e_1^- does not occur; since $e_1 <^- e_2$, e_1^- occurs before e_2^- in w or e_2^- does not occur; $e_1 <^- e_2$ shows that e_1^- does not belong to the last $co^{\pm}$-equivalence class, hence e_1^- does occur in w. Thus, the swapping of e_1 and e_2 in (w, l) is defined and leads to some (v, l). We have to show that (v, l) represents q, i.e. $e <' e'$ if and only if e^- occurs before e'^+ in v for all $e, e' \in E$.

The position of any e'^+ is the same in v and w. If $e_1 \neq e \neq e_2$, then the position of e^- is the same, too; hence, $e <' e'$ if and only if $e < e'$ (by definition of $swap(p)$) if and only if e^- occurs before e'^+ in w (since (w, l) represents p) if and only if e^- occurs before e'^+ in v. For the case $e=e_1$ we have: $e_1 <' e'$ if and only if $e_2 < e'$ if and only if e_2^- occurs before e'^+ in w if and only if e_1^- occurs before e'^+ in v (by definition of $swap(w, l)$). The case $e=e_2$ is analogous.

(ii) $\Rightarrow$ (iii) If $p=q$, we can choose $(w, l)=(v, l)$; hence let us assume that $p \neq q$. First let (v, l) be a closed representation of q, and furthermore by (ii) let (w', l) and (v', l) be

closed representations of p and q such that (v', l) can be obtained from (w', l) by swapping e_1 and e_2. (v, l) can be obtained from (v', l) by commuting starts and commuting ends; we can perform the same commutations in (w', l) to obtain some (w, l) where swapping e_1 and e_2 gives (v, l).

The only problem would arise if we wanted to commute e_1^+ and e_2^+ or e_1^- and e_2^-. In this case we would have some representation $(w', l) = (w_1 e_1^+ e_2^+ w_2, l)$ of p (or similarly for e_1^- and e_2^-) and swapping e_1 and e_2 would give a representation $(v', l) = (w_1 e_1^+ e_2^+ w_2', l)$ of q. Commuting e_1^+ and e_2^+ in (w', l) still gives a representation $(w_1 e_2^+ e_1^+ w_2, l)$ of p, but this representation is isomorphic to (v', l) by exchanging e_1 and e_2; hence $p = q$ (or, more precisely, p is isomorphic to q), contradicting our assumption.

It remains the case that (v, l) is not closed. Any given representation (v, l) of q can be obtained from a closed representation (v', l) by deleting ends from the end. Choose (w', l) with $(v', l) \in swap(w', l)$ and delete the corresponding ends from w' to obtain some representation (w, l) of p with $(v, l) \in swap(w, l)$.

(iii) $\Rightarrow$ (iv) obvious, since in (iii) we can choose (v, l) to be closed, which implies closedness of (w, l).

(iv) $\Rightarrow$ (v) obvious

(v) $\Rightarrow$ (i) Let (w, l) be a representation of $p = (E, <, l)$, and let (v, l) be a representation of $q = (E, <', l)$. If $(v, l) = (w, l)$, then $p = q$. Hence, let (v, l) be obtained from (w, l) by swapping e_1 and e_2. Since e_1^+ occurs before e_2^+ in w, we cannot have $e_2 <^+ e_1$, thus $e_1 <^+ e_2$ or $e_1^+ \; co \; {}^{\pm}e_2^+$. In the latter case, let w' be obtained by exchanging e_1^+ and e_2^+ in w. Since w' can also be obtained by commuting starts in w, we know that (w', l) also represents p. Now w' and v are isomorphic—the corresponding bijection simply exchanges e_1 and e_2. Hence, $p = q$ (or, more precisely, p and q are isomorphic).

Thus, we can assume $e_1 <^+ e_2$ which implies that not $e_2 < e_1$. Since e_2^+ occurs before e_1^- in w, we also do not have $e_1 < e_2$, and we conclude that $e_1 \; co \; e_2$. Since e_1^- occurs before e_2^- in w or e_2^- does not occur in w at all, we cannot have $e_2 <^- e_1$; thus, $e_1 <^- e_2$ or $e_1^- \; co^{\pm} \; e_2^-$. In the latter case, v can be obtained from w by commuting ends—or, if e_2^- does not occur in w, we have that v and w can be obtained from some closed representation of p by commuting ends and deleting ends from the end. Hence, $p = q$ in this case.

Thus, we can now assume $e_1 <^+ e_2$, $e_1 \; co \; e_2$ and $e_1 <^- e_2$, hence swapping of e_1 and e_2 in p is defined and gives some $q' \in swap(p)$. The proof that (v, l) represents q', hence $q \in swap(p)$, is the same as above. ∎

In the above proof, we have seen that applying the swap-operation to a labelled interval sequence might lead to one that represents the same labelled interval order—corresponding to the fact that labelled interval orders are more abstract than labelled interval sequences. This possibility is the reason for including the clauses "$p = q$" and "$(v, l) = (w, l)$" in the lemma.

In this context, it might be interesting to note that the swap-operation can be applied only finitely often. Let $(v, l) \in swap(w, l)$; let w^- be obtained from w by deleting all starts and similarly for v^-. Then w^- is lexicographically smaller than v^- with respect to the ordering $\prec$ defined by: $e_1^- \prec e_2^-$ if e_1^+ occurs before e_2^+ in w or, which is the same, in v. Thus, swapping can be applied to a labelled interval sequence only finitely often, and this carries over to labelled interval orders by Lemma 3.4. (Observe that for $q \in swap(p)$ we have $q \neq p$ by 3.2 and p and q cannot have a representation in common.)

As a consequence of Lemma 3.4, we see that swapping for labelled interval orders is well-defined:

PROPOSITION 3.5. *Let p be a labelled interval order and $q \in swap(p)$. Then q is a labelled interval order, too.*

Proof. Lemma 3.4 shows that q has a representation; hence it must be a labelled interval order. ∎

Furthermore, we can translate Lemma 3.4 to the case of iterated swapping.

THEOREM 3.6. *Let p be a labelled interval order. Then the following five statements are equivalent*:

(i) $q \in swap^*(p)$.

(ii) *For each representation (w, l) of p there exists some representation (v, l) of q with $(v, l) \in swap^*(p)$.*

(iii) *For each representation (v, l) of q there exists some representation (w, l) of p with $(v, l) \in swap^*(w, l)$.*

(iv) *There exists some closed representation (w, l) of p and some representation (v, l) of q such that $(v, l) \in swap^*(w, l)$.*

(v) *There exists some representation (w, l) of p and some representation (v, l) of q such that $(v, l) \in swap^*(w, l)$.*

Proof. (i) $\Rightarrow$ (ii), (i) $\Rightarrow$ (iii) and (v) $\Rightarrow$ (i) follow inductively from 3.4; (ii) $\Rightarrow$ (iv), (iii) $\Rightarrow$ (iv), and (iv) $\Rightarrow$ (v) are obvious.

We close this section by two results on the interplay of swapping with augmenting labelled interval orders and with the prefix-relation.

PROPOSITION 3.7. *Let p, q, and q' be labelled interval orders such that $q \in swap^*(p)$ and q' is an augmentation of q. Then there exists an interval augmentation p' of p with $q' \in swap^*(p')$.*

Proof. It is enough to consider the case that q is obtained from p by one swap-operation, say by swapping e_1

and e_2. Furthermore, by Theorem 2.9, it is enough to consider the case that q' is obtained by adding one pair to the ordering relation of q, say (e_3, e_4). By Lemma 2.10, q has a closed representation $(v_1 e_4^+ e_3^- v_2, l)$ and q' is represented by $(v_1 e_3^- e_4^+ v_2, l)$. By Theorem 3.6, there is a representation (w, l) of p such that swapping e_1 and e_2 in (w, l) gives $(v_1 e_4^+ e_3^- v_2, l)$.

If $\{e_1, e_2\}$ and $\{e_3, e_4\}$ have no event in common, it is obvious that $w = w_1 e_4^+ e_3^- w_2$, and $(w_1 e_3^- e_4^+ w_2, l)$ represents some augmentation p' of p such that swapping e_1 and e_2 in $(w_1 e_3^- e_4^+ w_2, l)$ is defined and yields $(v_1 e_3^- e_4^+ v_2, l)$, thus $q' \in swap^*(p')$. It is not hard to see that the same argument works for $e_4 \in \{e_1, e_2\}$ and $e_3 \notin \{e_1, e_2\}$.

If $e_4 \notin \{e_1, e_2\}$ but $e_3 \in \{e_1, e_2\}$, say $e_3 = e_1$, then $(w, l) = (w_1 e_4^+ e_2^- w_2, l)$ represents p, $(v_1 e_4^+ e_1^- v_2, l)$ represents q and $(v_1 e_1^- e_4^+ v_2, l)$ represents q'. Now $(w_1 e_2^- e_4^+ w_2, l)$ represents an augmentation p' of p, swapping of e_1 and e_2 is defined and it yields $(v_1 e_1^- e_4^+ v_2, l)$, i.e., $q' \in swap^*(p')$.

Finally, we have to consider the case $\{e_1, e_2\} = \{e_3, e_4\}$. Since after swapping e_1 and e_2 the starts and ends of these events have to appear in $v_1 e_4^+ e_3^- v_2$ in the order $e_1^+, e_2^+, e_2^-, e_1^-$, we would conclude that $e_4 = e_2 = e_3$; thus, this final case cannot occur. ∎

PROPOSITION 3.8. *Let p, q, and q' be labelled interval orders such that $q \in swap(p)$ and q' is a prefix of q. Then there exists a prefix p' of p such that $q' \in swap(p')$ or q' is an augmentation of p'.*

Proof. Let q be obtained by swapping e_1 and e_2 in p. There are several cases:

(a) Neither e_1 nor e_2 belongs to q'. Then q' is a prefix of p, too, and, we can choose $p' = q'$; q' is an augmentation of p' in this case.

(b) Both e_1 and e_2 belong to q'. Choose p' as the prefix of p which has the same events as q'. (Note that these events form a left-closed set in p, since an event has a predecessor in p which is not a predecessor in q only if this predecessor is e_1 or e_2, and both of these are in p'.) From Lemma 2.4 it is clear that we either can swap e_1 and e_2 in p' to obtain q' or e_1 and e_2 have the same successors in p'. In the latter case $p' = q'$.

(c) e_1, but not e_2 belongs to q'. Since e_2 is missing in q', all its successors in q are missing, too. These are the successors of e_1 in p and subsume the successors of e_2 in p by Lemma 2.4. Hence, e_1 has no successors in q' and q' is a prefix of p.

(d) e_2, but not e_1 belongs to q'. This is the most interesting case, since due to this case we really have to involve augmentation in 3.8. Let E' be the set of events of q'. Choose p' as the prefix of p with event set $E'' = E' - \{e_2\} \cup \{e_1\}$.

First, we have to check that E'' is left-closed in p. So let $e < e' \in E''$ in p. If $\{e, e'\} \cap \{e_1, e_2\} = \varnothing$, then $e < e' \in E'$ in q, hence $e \in E'$ and $e \in E''$. If $e' = e_1$, then e is also a predecessor of e_2 in p by 2.4, hence this holds in q, and $e \in E'$ and $e \in E''$ follow. Since $e_2 \notin E''$ and $e_1 \in E''$, it remains to check the case $e = e_2$. In this case, e' is a successor of e_2 in p, hence also of e_1 in q; we get the contradiction $e_1 \in E'$.

The ordering relations of p' and q' coincide except for e_1 and e_2. If $e < e_1$ in p', then $e < e_2$ in p by 2.4, hence also $e < e_2$ in q and q'. If $e_1 < e$ in p', then also in p; hence $e_2 < e$ in q and q'. This shows that if we isomorphically rename e_1 to e_2 in p', we get another representative of p', which has q' as an augmentation. Thus, q' is an augmentation of p'. ∎

Remark. (i) To see that, in Proposition 3.8, q' might indeed be a proper augmentation, consider p and q in Fig. 2. Choose q' as the prefix of q which has one a-, one b-, and one c-labelled event. The only possible prefix p' of p consists of the c-labelled event and its two predecessors.

(ii) We have seen that the operations of swapping and augmenting can just as well be performed on the level of representations as on the level of interval orders. It is interesting to note that such a correspondence does not hold for the prefix-relation. Consider again the prefix of p in Fig. 2 consisting of the c-labelled event and its two predecessors, and let (v, l) be a representation of this prefix. In (v, l), we obviously have only one start of a b-labelled event; furthermore, the end of this event occurs in v, since this end must occur before the start of the c-labelled event. Let (w, l) be any representation of p. Since in p there are two concurrent b-labelled events, we must have two starts of b-labelled events in (w, l) before the first end of such an event. Hence, v cannot be a prefix of w. Compare [Vog92] and the trunk-relation mentioned there.

4. PETRI NETS, INTERVAL SEMIWORDS, AND SPLITTING

In this section, a very brief introduction to Petri nets is given. For further information the reader is referred to, e.g., [Pet81, Rei85]. We will deal with safe Petri nets (place/transition-nets) whose transitions are labelled with actions from some infinite alphabet Σ or with the empty word λ. These actions are left uninterpreted; the labelling only indicates that two transitions with the same label from Σ represent the same action occurring in different internal situations, while λ-labelled transitions represent internal, unobservable actions. We assume that Σ also contains for each $a \in \Sigma$ and $i \in \mathbb{N}$ the action a_i.

Thus, a *labelled Petri net* $N = (S, T, W, lab, M_N)$ (or just a *net* for short) consists of finite disjoint sets S of *places* and T of *transitions*, the *weight function* $W: S \times T \cup T \times S \to \{0, 1\}$, the *labelling lab*: $T \to \Sigma \cup \{\lambda\}$, and the *initial marking* $M_N: S \to \mathbb{N}_0$. When we introduce a net N, then we assume that implicitly this introduces its components

S, T, ... In general, we will not distinguish isomorphic nets (just as for partial orders).

- A *multiset* over a set X is a functlon $\mu: X \to \mathbb{N}_0$. We identify $x \in X$ with the multiset that is 1 for x and 0 everywhere else. For multisets, multiplication with scalars from $\mathbb{N}_0$ and addition is defined elementwise.

- A *marking* is a multiset over S; a *step* is a multiset over T. A step μ is *enabled* under a marking M, denoted by $M[\mu\rangle$, if $\sum_{t \in \mu} \mu(t) \cdot W(s, t) \leqslant M(s)$ for all $s \in S$. If $M[\mu\rangle$ and $M'(s) = M(s) + \sum_{t \in \mu} \mu(t)(W(t, s) - W(s, t))$, then we denote this by $M[\mu\rangle M'$ and say that μ can *occur* or *fire* under M, yielding the *follower marking* M'. We also say that the transitions of μ can fire *concurrently*. Since transitions are special steps, this also defines $M[t\rangle$ and $M[t\rangle M'$ for $t \in T$.

- This definition of enabling and occurrence can be extended to sequences as usual: a sequence w of steps is *enabled* under a marking M, denoted by $M[w\rangle$, and yields the follower marking M' when *occurring*, denoted by $M[w\rangle M'$, if $w = \lambda$ and $M = M'$ or $w = w'\mu$, $M[w'\rangle M''$ and $M''[\mu\rangle M'$ for some marking M''. If w is enabled under the initial marking, then it is called a *transition step sequence*, or—in the case where $w \in T^*$—a *transition firing sequence*.

We can extend the labelling of a net to steps by $lab(\mu) = \sum_{t \in T,\, lab(t) \neq \lambda} \mu(t) \cdot lab(t)$, where the empty sum equals the empty word. Then we can extend the labelling also to sequences of steps or transitions as usual, i.e., homomorphically; note that internal actions are automatically deleted in the labelling of a sequence. Next, we lift the enabledness and firing definitions to the level of actions:

- A sequence v of multisets over Σ is *image enabled* under a marking M, denoted by $M[v\rangle\!\rangle$, if there is some w with $M[w\rangle$ and $lab(w) = v$. If $M = M_N$, then v is called an *action step sequence* or just *step sequence*; if $w \in T^*$, then v is called an *action firing sequence* or just *firing sequence*. We call two nets *step-sequence equivalent* if they have the same step sequences. We call two nets *language equivalent* if they have the same firing sequences.

- For a marking M the set $[M\rangle$ of markings *reachable from* M is defined as $\{M' \mid \exists w \in T^*: M[w\rangle M'\}$. A marking is called *reachable* if it is reachable from M_N. The net is *safe* if $M(s) \leqslant 1$ for all places s and reachable markings M.

- Two not necessarily distinct transitions t_1 and t_2 are concurrently enabled under some marking M if $M[t_1 + t_2\rangle$. A transition t is *self-concurrent* if $M[2t\rangle$ for some reachable marking M. An action $a \in \Sigma$ is *autoconcurrent* if $M[2a\rangle\!\rangle$ for some reachable marking M.

General Assumption. All nets considered in this paper are safe and without isolated transitions.

This implies that all nets in this paper are free of self-concurrency, but it does not exclude autoconcurrency.

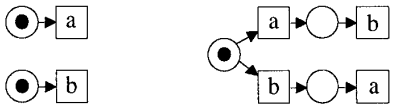


FIGURE 7

The purpose of studying "true concurrency" is to distinguish the independent execution of actions from their execution in arbitrary order. The language of a net only captures the latter view of concurrency; e.g., the nets of Fig. 7 are language equivalent, but obviously the first net can perform actions a and b independently of each other, while the second cannot. Petri net theory has a long tradition of studying "true concurrency" using partial orders. Most often a partial order semantics is given by so-called processes; see, e.g., [BF88]. In this approach the partial order models causality; consequently, the semantics of the nets in Fig. 7 are incomparable.

Another view is that concurrency is more than arbitrary interleaving but includes it. From this point of view the semantics of the first net of Fig. 7 should include the semantics of the second net. This idea is formalized in the (transition) partial word semantics of [Gra81], which coincides with the (transition) semiword semantics of [Sta81] for the nets we consider here since these are free of self-concurrency. A transition semiword of a net N is a partial order labelled with transitions of N; any set of pairwise concurrent elements of this partial order represents a step that can be fired provided the precedences prescribed by the partial order are observed.

- Let N be a net, M a marking of N, and $p = (E, <, l)$ a partial order labelled over T. We call p a *transition semiword* of N *enabled* under M if for all disjoint subsets B and C of E we have: If all elements of C are pairwise concurrent and B and $B \cup C$ are left-closed, then l is injective on C and the step $l(C)$ is enabled under $M + \sum_{e \in B} W(l(el), .) - W(., l(e))$, i.e.,

$$\forall s \in S: \sum_{e \in C} W(s, l(e)) \leqslant M(s) + \sum_{e \in B} W(l(e), s) - W(s, l(e)).$$

We write $M[p\rangle$. If p is enabled under M_N we simply say that p is a *transition semiword* of N.

Figure 8 shows a net and one of its transition semiwords; some sets B and C are indicated, and indeed, we can fire the step $t_3 + t_4$ if we fire t_1 and t_2 first.

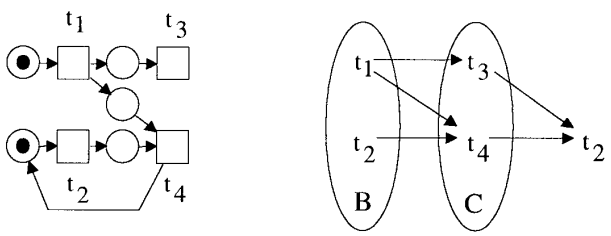


FIGURE 8

It is easy to see that the set of transition semiwords of a net is closed under augmentation; i.e., if p is an augmentation of a transition semiword q, then p is a transition semiword, too. This shows that concurrency in transition semiwords is just seen as a possibility; possibly concurrent transitions can also be performed sequentially.

For safe nets the so-called event structures of processes are just the least sequential transition semiwords, see [Kie88, Vog92]. In these transition semiwords, the ordering can be seen as modelling causality.

- The *image* q of a partial order $p=(E, <, l)$ labelled over T is $(E', <', l')$ where $E'=\{e\in E \mid lab(l(e))\neq\lambda\}$, $<'=<|_{E'\times E'}$ and $l'=lab\circ l|_{E'}$. If p is a transition semiword of N, then q is an *action semiword* or just a *semiword* of N. Two nets are *semiword equivalent* if they have the same semiwords.

Next, we combine the definitions of semiwords and interval orders.

- A transition semiword $(E, <, l)$ of a net N is an *interval transition semiword* of N if $(E, <)$ is an interval order; its image is an *interval action semiword* or an *interval semiword* for short. The set of interval semiwords of N is denoted by $ISW(N)$. Two nets are *interval-semiword equivalent* if they have the same interval semiwords.

Since the elements of an interval order correspond to closed intervals of real numbers, an interval semiword can intuitively be seen as the observation of a system run where each firing takes some time. Hence, they give a partial order semantics which is not so much related to causality, but rather has a temporal flavor.

Instead of the interval semiwords themselves, we can also consider their representations:

- An *interval word* of a net N is a representation of an interval semiword of N; the set of interval words is denoted by $IW(N)$. Two nets are interval-word equivalent if they have the same interval words.

Interval words of Petri nets can also be defined independently of interval semiwords, see [Vog92]; they are also called ST-traces, see [Gla90]. Since a labelled interval order can be reconstructed from any of its representations, interval-word and interval-semiword equivalence obviously coincide.

The operation of action refinement, see [Vog92] (also for additional references), is meant to support the hierarchical design of systems. An action refinement *ref* assigns to each action $a\in\Sigma$ a subsystem $ref(a)$. From a net N, we obtain the net $ref(N)$ by replacing each a-labelled transition by a separate copy of the net $ref(a)$.

If one is interested in action refinement, one should use a semantics that supports this operation in the following way: it should be possible to determine the semantics of $ref(N)$ from the refinement *ref* and the *semantics* of N, i.e., without knowing N itself. Or equivalently, we require: if nets are equivalent with respect to the semantics in use, then the refined nets should be equivalent again, i.e., the equivalence induced by the semantics should be a congruence with respect to action refinement. Process, semiword, and interval-semiword equivalence are such congruences [Vog91, Vog92]. Only interval-semiword equivalence has the following additional property: if nets N and N' are not interval-semiword equivalent, then there exists a refinement *ref* such that $ref(N)$ and $ref(N')$ are not language equivalent. Hence, if we take the language as a starting point, then interval semiwords make exactly those additional distinctions between nets which are necessary to get a congruence. Expressed in a technical way, interval-semiword equivalence is *fully abstract* with respect to language equivalence and action refinement, i.e., it is the coarsest congruence for action refinement that is finer than language equivalence.

In this paper, we are only interested in a very simple, but natural sort of refinement, namely in splitting. In such a refinement, an action is performed in several phases, i.e., we replace $a\in\Sigma$ with a sequence $a_1a_2\cdots a_n$. Splitting is one way to express that an action has a duration. Note that we do not refine internal actions.

DEFINITION 4.1. Let N be a net, $n\in\mathbb{N}$. Then $split_n(N)=(S_n, T_n, W_n, lab_n, M_{split_n(N)})$, the *splitting* of N into n phases, is defined by

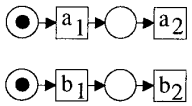
$$S_n = S \dot\cup \{(s_i, t) \mid i=1, ..., n-1, t\in T \text{ with } lab(t)\neq\lambda\}$$

$$T_n = \{(t, j) \mid t\in T \text{ with } lab(t)\neq\lambda,\ j=1, ..., n\} \dot\cup \{t\in T \mid lab(t)=\lambda\}$$

W_n has value 1 for the following pairs (and 0 otherwise)

$((t, i), (s_i, t))$	$i=1, ..., n-1$		
$((s_i, t), (t, i+1))$	$i=1, ..., n-1$		
$(s, (t, 1))$	if $W(s, t)=1$		
$((t, n), s)$	if $W(t, s)=1$		
(s, t)	if $W(s, t)=1$	and	$lab(t)=\lambda$
(t, s)	if $W(t, s)=1$	and	$lab(t)=\lambda$
$lab_n(t, j)=lab(t)_j$			
$lab_n(t)=\lambda$	if $lab(t)=\lambda$		
1	if $s\in S$	and	$M_N(s)=1$

$$M_{split_n(N)}(s)=\begin{cases}1 & \text{if } s\in S \text{ and } M_N(s)=1\\ 0 & \text{otherwise.}\end{cases}$$



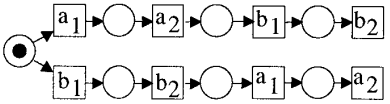


FIGURE 9

We call nets N, N' *$split_n$-language equivalent* or simply *$split_n$-equivalent* if $split_n(N)$ and $split_n(N')$ are language equivalent.

Figure 7 shows two nets that are language equivalent or, which is the same, $split_1$-equivalent; but they are not $split_2$-equivalent, since only the net on the left can perform $a_1 b_1 a_2 b_2$ after splitting. Figure 9 shows the results of applying $split_2$.

Figure 10 shows two nets that are $split_2$- but not $split_3$-equivalent. These nets are not $split_3$-quivalent, since only the net on the left can perform $w = b_1 b_2 a_1 a_2 a_3 b_1 b_3 c_1$ after application of $split_3$. It is not so easy to see that they are $split_2$-equivalent; but it is feasible to check all possible firing sequences of the split nets. (Observe that the net on the left can essentially perform the semiword on the left of Fig. 11, while the net on the right can perform one of the semiwords in the middle or on the right of Fig. 11.) The essential point is that both nets can perform $b_1 a_1 a_2 b_1 b_2 c_1$ after application of $split_2$, which is somewhat similar to w above. Note that b can be performed autoconcurrently; without autoconcurrency, $split_2$- and $split_3$-equivalence coincide.

More generally, it has been shown in [GV95] that for all $n \in \mathbb{N}$ there are nets which are split$_n$- but not $split_{n+1}$-equivalent. On the other hand, it is not difficult to see that $split_{n+1}$-equivalence implies $split_n$-equivalence: for some net N and $n \in \mathbb{N}$, consider a firing sequence of $split_{n+1}(N)$ where each occurrence of any a_n, $a \in \Sigma$, is immediately succeeded by a_{n+1}. If the a_n corresponds to some (t, n), then after firing (t, n) the only enabled a_{n+1}-labelled transition is $(t, n+1)$. Hence, if we contract each subsequence $a_n a_{n+1}$ to a_n, we get a firing sequence of $split_n(N)$. Vice versa, each firing sequence of $split_n(N)$ can be obtained this way. We conclude:

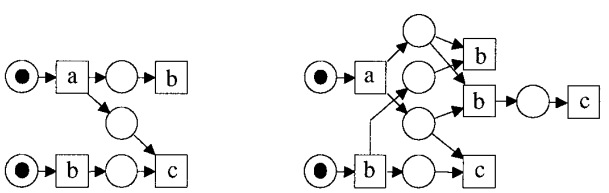


FIGURE 10

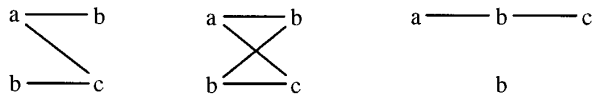


FIGURE 11

THEOREM 4.2. *For all $n \in \mathbb{N}$, $split_{n+1}$-equivalence implies $split_n$-equivalence, but not the other way round.*

Remark. We consider only split-refinements where each visible action is split into the same number of phases. This is no restriction for the following reason: if all actions of N are split into at most n phases, we can obtain the language of the split net from the language of $split_n(N)$ in much the same way as above by contracting $a_k \cdots a_n$ to a_k for an action a that is split into $k < n$ phases. Hence, $split_n$-equivalence implies equivalence with respect to arbitrary splittings into at most n phases.

5. A NEW CONGRUENCE FOR SPLITTING

From the general result that interval-semiword equivalence is fully abstract for language equivalence and action refinement, it is clear that, in particular, it is a congruence for splitting. In this paper, we want to determine the coarsest such congruence, i.e., one that is fully abstract with respect to splitting and language equivalence. In other words, we want to determine the limit of the sequence of the increasingly finer $split_n$-equivalences.

On the level of bisimulation, a corresponding result has been shown. ST-bisimulation is a variant of bisimulation that corresponds to interval (semi-)words. ST-bisimilarity is fully abstract with respect to action refinement and bisimilarity [Gla90, Vog93], and it is also fully abstract with respect to splitting and bisimilarity [GL91].

On the language level, it will turn out that such a coincidence does not hold. Consider the net N of Fig. 12; it can perform the interval semiword in the middle of Fig. 12, and we can conclude this from the firing sequence $a_1 b_1 b_2 a_2 a_3 b_1 b_3 c_1$ of $split_3(N)$. This sequence shows that some b starts independently of some a; after a, but not after the first b, a second b starts. The essential point is that b_3, which comes next, must be the end of the first b, since the second b has only performed its first phase; hence, c starts after a and the first b, but independently of the second b. If we try to construct a similar firing sequence for the labelled interval order on the right, we start with $a_1 b_1 b_2 a_2 a_3 b_1$ as

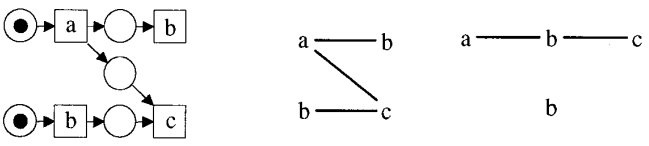


FIGURE 12

before; but now we have to continue with $b_2b_3c_1$. In the resulting sequence, we cannot see anymore which b ends before c. In fact, $a_1b_1b_2a_2a_3b_1b_2b_3c_1 \in split_3(N)$ although the labelled interval order on the right is not an interval semiword of N. Instead, it is obtained from the interval semiword in the middle by swapping; this observation motivates the new semantics, which is defined by closing $ISW(N)$ under swapping. (Similarly, transition semiwords can be obtained by closing the set of event structures of processes under augmentation, as remarked above.)

DEFINITION 5.1. The *swap-interval-semiword semantics* of a net N is the set *swap-ISW*(N) of all labelled interval orders q for which there exists some $p \in ISW(N)$ with $q \in swap^*(p)$. We also call the elements of *swap-ISW*(N) *swap-semiwords*. If we have *swap-ISW*$(N) =$ *swap-ISW*(N') for nets N and N', we call these nets *swap-semiword equivalent* or simply *swap equivalent*.

The proof that swap equivalence is the fully abstract congruence we are looking for is based on interval sequences. Hence, as a first step, we characterize swap equivalence with these sequences

DEFINITION 5.2. The *swap-interval-word semantics* of a net N is the set *swap-IW*(N) of all labelled interval sequences (v, l) for which there exists some $(w, l) \in IW(N)$ with $(v, l) \in swap^*(w, l)$. We also call the elements of *swap-IW*(N) *swap-words*. If *swap-IW*$(N) =$ *swap-IW*(N') for nets N and N', we call these nets *swap-word equivalent*.

THEOREM 5.3. (i) *For a net N, swap-IW(N) is the set of all representations of elements of swap-ISW(N).*

(ii) *Swap-word and swap-semiword equivalence coincide.*

Proof. (i) By Theorem 3.6(i) $\Rightarrow$ (ii), a representation of some $q \in$ *swap-ISW*(N) is an element of *swap-IW*(N). By Theorem 3.6(v) $\Rightarrow$ (i), each element of *swap-IW*(N) represents some $q \in$ *swap-ISW*(N).

(ii) immediate from (i).

Lemma 5.4.12 in [Vog92] shows how, for any refinement *ref*, we can determine the interval words of some *ref*(N) from the interval words of N. Here, we are only interested in splitting instead of general refinement, and we are only interested in the language of the split net, not in its interval words. Accordingly, we adapt Lemma 5.4.12 in [Vog92] to our setting and our slightly different notation.

DEFINITION 5.4. Let (w, l) be a labelled interval sequence over E, $n \geqslant 2$. A *concrete n-refinement* of (w, l) is a sequence $v \in (\Sigma \times E)^*$ such that:

— If (b, e) occurs in v, then $b = a_k$ for $a = l(e)$ and some $k \leqslant n$.

— Define a morphism $proj\colon (\Sigma \times E)^* \to (E^{\pm})^*$ by

$$proj(b, e) = \begin{cases} e^+ & \text{if } b = a_1 \\ e^- & \text{if } b = a_n \\ \lambda & \text{otherwise.} \end{cases}$$

Then $proj(v) = w$.

— For each $e \in E$, define a morphism $proj_e\colon (\Sigma \times E)^* \to \Sigma^*$ by

$$proj_e(b, e') = \begin{cases} b & \text{if } e = e' \\ \lambda & \text{if } e \neq e'. \end{cases}$$

Then $proj_e(v) = a_1a_1 \cdots a_k$ for some $k \leqslant n$.

An *abstract n-refinement* of (w, l) is some $abs(v)$, where v is a concrete n-refinement of (w, l) and $abs\colon (\Sigma \times E)^* \to \Sigma^*$ is a morphism defined by $abs(b, e) = b$.

The condition $proj(v) = w$ says that the first and the last phases in v exactly match the starts and ends in w. The condition on $proj_e(v)$ shows that e is in v subdivided into a suitable sequence of phases. In a concrete n-refinement, the second component tells us which a_i are phases of the same occurrence of action a. This is an information we do not have in $abs(v)$, and neither in a firing sequence of $split_n(N)$. Formally, *abs* depends on the event set E and should consequently be indexed with E. We will use *abs* to denote any of these functions.

With these definitions, Lemma 5.4.12 of [Vog92] gives:

PROPOSITION 5.5. *Let N be a net, $n \geqslant 2$. Then $L(split_n(N))$ is the set of all abstract n-refinements of interval words of N.*

With this result, we will show that swap-equivalence implies $split_n$-equivalence, the first half of our main result. The proof consists essentially of the following lemma.

LEMMA 5.6. *Let (w, l) be a labelled interval sequence, $(w', l) \in swap(w, l)$, and v an abstract n-refinement of (w', l) for some $n \geqslant 2$. Then v is an abstract n-refinement of (w, l), too.*

Proof. Let (w', l) be obtained from (w, l) by swapping e_1 and e_2; i.e., $e_1^+, e_2^+, e_1^-, e_2^-$ occur in w in this order (or e_2^- does not occur at all) while in w' they occur in the order $e_1^+, e_2^+, e_2^-, e_1^-$ (or e_1^- does not occur at all). Furthermore, e_1 and e_2 have the same label, say a.

Let v_1' be a concrete n-refinement of (w', l) with $abs(v_1') = v$. By the above (a_1, e_1), (a_1, e_2), (a_n, e_2) and (a_n, e_1) occur in v_1' in this order (or (a_n, e_1) does not occur at all). For each prefix of v_1', consider the pair (j, k) where j, k resp., is the largest index i of some (a_i, e_1), (a_i, e_2) resp., occurring in the prefix (or 0 if otherwise undefined). Considering the prefixes one after the other, these pairs increase

monotonously from $(0, 0)$ over $(1, 0)$ and $(j, 1)$ for some $j \geq 1$ to reach some (j, n) with $j < n$; thus, they must have some value (k, k) inbetween. Hence, we can find v'_2 with $v'_1 = v'_2 v'_3$ and $k < n$ such that (a_k, e_1) and (a_k, e_2) occur in v'_2 but neither (a_{k+1}, e_1) nor (a_{k+1}, e_2). If we construct v_3 from v'_3 by replacing each (a_l, e_1) by (a_l, e_2) and vice versa, $l > k$, then $v_1 = v'_2 v_3$ is a concrete n-refinement of (w, l) with $abs(v_1) = v$. The result follows. ▮

LEMMA 5.7. *Let N be a net, $n \geq 2$. Then $L(split_n(N))$ is the set of all abstract n-refinements of swap-interval-words of N.*

Proof. By 5.5, this set of abstract n-refinements contains $L(split_n(N))$. Vice versa, it follows from 5.6 that an abstract n-refinement of a swap-interval-word of N is also one of an interval word of N, hence in $L(split_n(N))$. ▮

THEOREM 5.8. *Swap-equivalence implies $split_n$-equivalence for all $n \in \mathbb{N}$.*

Proof. The result follows directly from 5.7 and 4.2. (Recall that swap-equivalence is the same as swap-interval-word equivalence). ▮

The reverse implication is a little harder to prove. For this proof, we have to add some information to an interval sequence and to extend some notions to this detailed version of interval sequences. We leave this proof for the next section, and just state the main result:

THEOREM 5.9. (i) *Nets are swap-equivalent if and only if they are $split_n$-equivalent for all $n \in \mathbb{N}$.*

(ii) *Swap-equivalence is fully abstract with respect to language equivalence and splitting.*

(iii) *Swap-equivalence is fully abstract with respect to language equivalence and $split_2$.*

Proof. (i) Theorems 5.8 and 6.8.

(ii) This follows from (i) once we have shown that swap-equivalence is a congruence for splitting. Hence, let nets N_1, N_2 be swap-equivalent and let $k \in \mathbb{N}$. The nets $split_n(split_k(N_m))$ and $split_{n \cdot k}(N_m)$, $m = 1, 2$, are isomorphic, except that the actions $(a_i)_j$ of $split_n(split_k(N_m))$, where $i = 1, ..., k$ and $j = 1, ..., n$, have to be renamed bijectively to $a_{(i-1) \cdot n + j}$, which are the actions of $split_{n \cdot k}(N_m)$. By (i), $split_{n \cdot k}(N_1)$ and $split_{n \cdot k}(N_2)$ are language equivalent. Therefore, $split_k(N_1)$ and $split_k(N_2)$ are $split_n$-equivalent for all $n \in \mathbb{N}$, and by (i) they are swap-equivalent.

(iii) Similarly, observe that a congruence for $split_2$ is a congruence for splitting in any 2^k, hence for arbitrary splitting. ▮

In order to check swap-equivalence, it seems that we have to consider all interval semiwords of a net and all labelled interval orders that can be obtained from them by iterated swapping. If we want to check an example as the one given in the introduction, such considerations are not feasible. The following observation helps a little.

THEOREM 5.10. *If, for nets N_1 and N_2, $ISW(N_1) \subseteq swap\text{-}ISW(N_2)$, then we have $swap\text{-}ISW(N_1) \subseteq swap\text{-}ISW(N_2)$.*

Proof. For $p_1 \in swap\text{-}ISW(N_1)$ there exists by definition some $p_2 \in ISW(N_1)$ with $p_1 \in swap^*(p_2)$. By assumption, $p_2 \in swap\text{-}ISW(N_2)$ and, hence, there exists $p_3 \in ISW(N_2)$ with $p_2 \in swap^*(p_3)$. We conclude $p_1 \in swap^*(p_3)$ and $p_1 \in swap\text{-}ISW(N_2)$. ▮

Theorem 5.10 shows that in order to prove swap-equivalence it suffices to check that all interval semiwords of one net are contained in the *swap-ISW* semantics of the other (and vice versa). But we can do better than this, as the next two theorems show.

THEOREM 5.11. (i) *For a net N, $swap\text{-}ISW(N)$ is closed under interval augmentation; i.e., if p is an interval augmentation of $q \in swap\text{-}ISW(N)$, then $p \in swap\text{-}ISW(N)$.*

(ii) *Let N_1 and N_2 be nets such that all least sequential interval semiwords of N_1 are contained in $swap\text{-}ISW(N_2)$. Then $swap\text{-}ISW(N_1) \subseteq swap\text{-}ISW(N_2)$.*

Proof. (i) Let $q \in swap\text{-}ISW(N)$; i.e., there exists some $p \in ISW(N)$ with $q \in swap^*(p)$. Let q' be an interval augmentation of q. By Proposition 3.7, there exists an interval augmentation p' of p with $q' \in swap^*(p')$. Since $ISW(N)$ is closed under interval augmentation, we conclude that $p' \in ISW(N)$ and $q' \in swap\text{-}ISW(N)$.

(ii) By assumption and (i), $ISW(N_1) \subseteq swap\text{-}ISW(N_2)$ and the result follows from 5.10. ▮

THEOREM 5.12. (i) *For a net N, $swap\text{-}ISW(N)$ is closed under taking prefixes.*

(ii) *Let N_1 and N_2 be nets such that each least sequential interval semiword p of N_1 is a prefix of an element of $swap\text{-}ISW(N_2)$. Then $swap\text{-}ISW(N_1) \subseteq swap\text{-}ISW(N_2)$.*

Proof. (i) Let q' be a prefix of $q \in swap\text{-}ISW(N)$, i.e., there exist $p_1 \in ISW(N)$ and $p_2, ..., p_n$ such that $p_{i+1} \in swap(p_i)$, $i = 1, ..., n-1$, and $p_n = q$. By Proposition 3.8 we can find a sequence $p'_1, ..., p'_n$ of labelled interval orders such that each p'_i is a prefix of p_i, $p'_n = q'$ and, for $i = 1, ..., n-1$, $p'_{i+1} \in swap(p'_i)$ or p'_{i+1} is an augmentation of p'_i. Since $ISW(N)$ is closed under taking prefixes, we have $p'_1 \in ISW(N) \subseteq swap\text{-}ISW(N)$. Since $swap\text{-}ISW(N)$ is by definition closed under swapping and by Theorem 5.11 closed under augmentation, we conclude that all p'_i—including q'—are in $swap\text{-}ISW(N)$.

(ii) follows from (i) and 5.11(ii).

In order to show the usefulness of these results, we apply them to the nets discussed in the introduction. For this application, we also need the relation between processes and semiwords. We do not define processes here and refer the reader to, e.g., [Vog92]; but hopefully, the following discussion will be intuitively clear also for those readers who do not know processes.

Processes describe runs of Petri nets. A process consists of events, representing transition firings, and of conditions, representing tokens that appear during a run of a net. A process induces a partial order on the events; if we label the events with the corresponding transitions, this labelled partial order is the *event structure* of the process. If we change the labelling to the corresponding actions and delete events that correspond to internal actions, then we obtain the *action structure* of the process. This action structure shows the actions that occurred and their causal relations. For example, the net N of Fig. 1 has the labelled partial order p of Fig. 2 and all the prefixes of p as action structures. The net N' in Fig. 1 additionally has q and its prefixes as action structures.

Action structures subsume all least sequential semiwords of a net; hence their augmentations subsume all semiwords, in particular all interval semiwords and all least sequential interval semiwords.

Using this result, we can show that N and N' are swap-equivalent. Since N is contained in N', it is quite clear that $ISW(N) \subseteq ISW(N')$, hence we have $swap\text{-}ISW(N) \subseteq swap\text{-}ISW(N')$. For the reverse containment, we want to apply Theorem 5.12(ii), and for this we want to construct all least sequential interval semiwords of N' from the action structures of N'.

All action structures of N' that do not involve the additional c-transition of N' are also action structures of N and nothing has to be checked. The other action structures of N' are q and the sequence abc. Both are already labelled interval orders, hence these subsume the least sequential interval semiwords of N' that we have to check. The sequence abc is a prefix of q, so by Theorem 5.12(ii) it is enough to check q. For q we have $q \in swap^*(p)$ and $p \in ISW(N)$; hence $q \in swap\text{-}ISW(N)$ and by 5.12(ii) $swap\text{-}ISW(N') \subseteq swap\text{-}ISW(N)$. We conclude that N and N' are indeed swap-equivalent.

THEOREM 5.13. (i) *Interval-semiword equivalence strictly implies swap-equivalence.*

(ii) *Swap-equivalence strictly implies split$_2$-equivalence.*

(iii) *Split$_2$-equivalence strictly implies step-sequence equivalence.*

Proof. (i) Since $swap\text{-}ISW(N)$ can be determined from $ISW(N)$ by definition, the implication is clear. Strictness follows from the example above, since q is an interval semiword of N', but not of N.

(ii) follows from Theorems 5.8 and 4.2.

(iii) The proof for the implication is similar to the proofs of 5.4.7 and 5.4.8 in [Vog92]. Strictness follows from the example discussed in [GG89]. ∎

Thus, we have placed swap-equivalence in the interleaving–"true concurrency" spectrum and we have shown how it can possibly be checked by hand for small examples. To decide swap-equivalence in general, it seems to be more promising to use the (sequential) representations instead of the labelled interval orders themselves; but so far, no decision algorithm is known.

6. SWAP-EQUIVALENCE IS THE LIMIT

In this section, we will prove the missing half of Theorem 5.9(i). As already announced, we have to consider a more detailed version of interval sequences.

DEFINITION 6.1. An *n-phase interval sequence* (w, l, f) *(over E)* consists of a labelled interval sequence (w, l) over E and a function $f: E \to \{1, ..., n\}$ such that $f(e) = n$ if and only if e^- occurs in w; for $e' \notin E$ we consider $f(e')$ as being equal to 0. Again, we usually identify isomorphic n-phase interval sequences; compare Section 2. The *length* of (w, l, f) is $\sum_{e \in E} f(e)$. Let (w_1, l_1, f_1) and (w_2, l_2, f_2) be n-phase interval sequences over E_1 and E_2. Then, (w_1, l_1, f_1) is a *prefix* of (w_2, l_2, f_2) if w_1 is a prefix of w_2, l_2 equals l_1 on E_1 and $f_1(e) \leqslant f_2(e)$ for all $e \in E_1$.

Intuitively, an n-phase interval sequence is a labelled interval sequence where the actions have n phases. The function f assigns to each event the number of corresponding action phases that have been completed. If $f(e) = n$, then all phases have been completed, e has ended, and consequently we must have e^- in w. The length counts the number of completed phases. With this intuition, the prefix-definition should be natural.

DEFINITION 6.2. We can obtain (w, l, f) from an n-phase interval sequence (w', l, f') by *swapping* e_1 and e_2, denoted by $(w, l, f) \in swap(w', l, f')$, if

$$l(e_1) = l(e_2),\ f(e) = \begin{cases} f'(e_1) & \text{if } e = e_2 \\ f'(e_2) & \text{if } e = e_1 \\ f'(e) & \text{otherwise} \end{cases}$$

and one of the following cases applies:

— e_1^+, e_2^+, e_1^- and e_2^- occur in w' in this order; w is obtained from w' by exchanging e_1^- and e_2^- (in this case $f = f'$).

— e_1^+, e_2^+ and e_1^- occur in w' in this order, but e_2^- does not occur; w is obtained from w' by replacing e_1^- by e_2^-.

— e_1^+ and e_2^+ occur in w' in this order, but e_1^- and e_2^- do not occur, and $f'(e_2) < f'(e_1)$; $w = w'$.

If (w, l, f) is obtained from (w', l, f') by swapping several (including zero) times, we write $(w, l, f) \in swap^*(w', l, f')$. ∎

Obviously, swapping yields another n-phase interval sequence. Furthermore, swapping as in 6.2 extends swapping of labelled interval sequences:

LEMMA 6.3. *If* $(w, l, f) \in swap^*(w', l, f')$ *for an n-phase interval sequence* (w', l, f') *over E, then*:

(i) $(w, l) \in swap^*(w', l)$;

(ii) *f and f' have the same values* (*also with the same multiplicities*) *on each set* $E_a = \{e \in E \mid l(e) = a\}$, $a \in \Sigma$.

Proof. (i) In 6.2, swapping is defined on the (w', l)-part as for labelled interval sequences except for the third case. In this case $(w', l) = (w, l)$ and hence $(w, l) \in swap^*(w', l)$, too.

(ii) Obvious.

Swapping and the prefix-relation are compatible in the following way:

LEMMA 6.4. *Let* (w_1, l_1, f_1), (w_1', l_1, f_1') *and* (w_2', l_2, f_2') *be n-phase interval sequences such that* $(w_1, l_1, f_1) \in swap^*(w_1', l_1, f_1')$ *and* (w_1', l_1, f_1') *is a prefix of* (w_2', l_2, f_2'). *Then* (w_1, l_1, f_1) *is a prefix of some* $(w_2, l_2, f_2) \in swap^*(w_2', l_2, f_2')$.

Proof. It is enough to consider $(w_1, l_1, f_1) \in swap(w_1', l_1, f_1')$ and to apply induction afterwards. Therefore, let (w_1, l_1, f_1) be obtained from (w_1', l_1, f_1') by swapping e_1 and e_2.

We have to check the three cases of Definition 6.2 separately; in each case, we will obtain (w_2, l_2, f_2) from (w_2', l_2, f_2') by swapping e_1 and e_2 or, in the third case where neither e_1^- nor e_2^- occurs in $w_1' = w_1$, we might choose (w_2, l_2, f_2) equal to (w_2', l_2, f_2'). In all cases, it is clear that: w_1 is a prefix of w_2 just as w_1' is a prefix of w_2'; all labellings coincide on those events occurring in w_1; for all $e \notin \{e_1, e_2\}$, $f_1(e) = f_1'(e) \leqslant f_2'(e) = f_2(e)$; furthermore, $(w_2, l_2, f_2) \in swap^*(w_2', l_2, f_2')$.

It remains to check the phase-values of e_1 and e_2 and the applicability of swapping for (w_2', l_2, f_2'). If we can apply swapping, then $f_1(e_1) = f_1'(e_2) \leqslant f_2'(e_2) = f_2(e_1)$ and analogously for e_2.

If the first case of 6.2 applies for the swapping of e_1 and e_2 in (w_1', l_1, f_1'), then we can also swap e_1 and e_2 in (w_2', l_2, f_2') by the first case of 6.2.

If the second case of 6.2 applies, then e_2^- might occur in w_2' such that we can swap e_1 and e_2 in (w_2', l_2, f_2') by the first case of 6.2; or e_2^- does not occur in w_2' and the second case of 6.2 can be applied.

Finally, if the third case of 6.2 applies for the swapping in (w_1', l_1, f_1'), then there are two subcases.

(a) e_1^- occurs in w_2' before e_2^-, it occurs in w_2' without e_2^- occuring, or neither e_1^- nor e_2^- occur in w_2'. Then we can swap e_1 and e_2 in (w_2', l_2, f_2') by the first, second or third case respectively of 6.2.

(b) e_2^- occurs in w_2', either before or without e_1^-. Then we choose (w_2, l_2, f_2) equal to (w_2', l_2, f_2'). We have $f_1(e_1) = f_1'(e_2) < f_1'(e_1) \leqslant f_2'(e_1) = f_2(e_1)$; $f_2(e_2)$ equals the maximal possible value n, hence $f_1(e_2) \leqslant f_2(e_2)$. ∎

Next, we define n-refinements of n-phase interval sequences. Here, the role of the function f should become clearer.

DEFINITION 6.5. A *concrete n-refinement* v of an n-phase interval sequence (w, l, f) over E is a concrete n-refinement of (w, l) such that, for all $e \in E$, $f(e)$ is the maximal k for which some (a_k, e) appears in v. If v is a concrete n-refinement of (w, l, f), then $abs(v)$ is an *abstract n-refinement* of (w, l, f).

The following connection to Definition 5.4 is obvious.

LEMMA 6.6. *Let* (w, l) *be a labelled interval sequence. Then v is a concrete* (*abstract*) *n-refinement of* (w, l) *if and only if v is a concrete* (*abstract*) *n-refinement of some n-phase interval sequence* (w, l, f).

The following lemma is crucial for the proof we give in this section. It shows how we can deduce some interval semiword of a net N from the language of some $split_n(N)$—at least up to swapping in a certain sense.

LEMMA 6.7. *Let* (w, l, f) *be an n-phase interval sequence over some set E such that* $n > |E|$. *Then there exists a concrete n-refinement v of* (w, l, f) *such that for any concrete n-refinement v' of any* (w', l', f'), *we have that* $abs(v) = abs(v')$ *implies* $(w, l, f) \in swap^*(w', l', f')$.

Proof. It is no restriction to consider only n-phase interval sequences (w', l', f') over the same set E as for (w, l, f) and with the same labelling l, since $|E|$ is the number of occurrences of a_1, $a \in \Sigma$, in $abs(v)$ and since we are interested in n-phase interval sequences only up to isomorphism.

The proof is by induction on the length of (w, l, f), where the case of length 0, i.e., $w = v = \lambda$, is clear. Thus, let (w, l, f) of non-zero length be given; we will choose a suitable prefix (w_1, l_1, f_1) in order to apply induction. There are several cases to consider. Fig. 13 might help to follow the constructions; the arrows indicate the prefix relation. We will look at two examples of the construction afterwards.

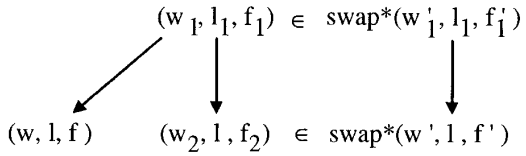


FIGURE 13

(a) There is some $e \in E$ with $1 < f(e) < n$ such that, for all $e_1 \in E$, $f(e_1) = f(e) - 1$ implies that e_1^+ occurs in w before e^+.

In this case, we choose $w_1 = w$, $l_1 = l$ and f_1 according to

$$f_1(e_1) = \begin{cases} f(e) - 1 & \text{if } e_1 = e \\ f(e_1) & \text{otherwise.} \end{cases}$$

Obviously, (w_1, l_1, f_1) is an n-phase interval sequence and we can find a corresponding v_1 by induction. Extend v_1 to $v = v_1 \cdot (l(e)_{f(e)}, e)$; obviously, v is a concrete n-refinement of (w, l, f).

Assume that we are given some (w', l, f') and v' with $abs(v) = abs(v')$. Then, we have $v' = v_1'(l(e)_{f(e)}, e')$ for some e' with $l(e') = l(e)$. If we define a prefix (w_1', l_1, f_1') of (w', l, f') by $w_1' = w'$ and

$$f_1'(e_1) = \begin{cases} f'(e') - 1 & \text{if } e_1 = e' \\ f'(e_1) & \text{otherwise,} \end{cases}$$

then v_1' is a concrete n-refinement of (w_1', l_1, f_1') with $abs(v_1) = abs(v_1')$. By induction we have $(w_1, l_1, f_1) \in swap^*(w_1', l_1, f_1')$. By Lemma 6.4, (w_1, l_1, f_1) is a prefix of some $(w_2, l, f_2) \in swap^*(w', l, f')$. Since f_2 and f' have the same values, cf. 6.3(ii), (w_1, l_1, f_1) and (w_2, l, f_2) coincide except for one value of the phase-functions: for e_2, say, we have $f_2(e_2) = f_1(e_2) + 1$. Furthermore, for e_2, we have $f_1(e_2) = f_1(e)$ and $l(e_2) = l(e)$.

Hence, if $e = e_2$, we conclude that $(w, l, f) = (w_2, l, f_2) \in swap^*(w', l, f')$. Otherwise, (w, l, f) and (w_2, l, f_2) coincide except that $f_2(e_2) = f_1(e_2) + 1 = f_1(e) + 1 = f(e)$ and $f_2(e) = f_1(e) = f_1(e_2) = f(e_2)$; this implies $f_2(e_2) = f_1(e) + 1 > f_2(e)$ and $f(e_2) = f_1(e) = f(e) - 1$. By choice of e, e_2^+ occurs in $w = w_2$ before e^+; since $f_2(e_2) > f_2(e)$, we can swap e_2 and e in (w_2, l, f_2) in order to obtain (w, l, f). In both cases the result follows.

(b) (a) is not the case.

In particular, if e_m is the event that starts last in w, then $f(e_m) \in \{1, n\}$. We have the following two subcases.

(b1) $f(e_m) = 1$ and $w = w_1 e_m^+$.

The interval sequence w_1 is defined over $E_1 = E - \{e_m\}$. If we restrict l and f to E_1, we obtain l_1 and f_1 such that (w_1, l_1, f_1) is a prefix of (w, l, f) and an n-phase interval sequence with $n > |E| > |E_1|$. By induction we find a suitable v_1, and we define $v = v_1(l(e_m)_1, e_m)$; obviously, v is a concrete n-refinement of (w, l, f).

Assume that we are given some (w', l, f') over E and v' with $abs(v) = abs(v')$. Then we have $v' = v_1'(l(e_m)_1, e)$ for some e and we may assume that $e = e_m$ (otherwise, exchange e and e_m in (w', l, f')). Hence, $w' = w_1' e_m^+$; we restrict f' to E_1 to get f_1'. Now (w_1', l_1, f_1') is a prefix of (w', l, f') with concrete n-refinement v_1'. By induction, $(w_1, l_1, f_1) \in swap^*(w_1', l_1, f_1')$. By Lemma 6.4, there is some $(w_2, l, f_2) \in swap^*(w', l, f')$ which has (w_1, l_1, f_1) as a prefix. We have $w_2 = w_1 e_m^+$ and get $(w, l, f) = (w_2, l, f_2)$.

(b2) Neither (a) nor (b1) is the case.

In this case, w has the form $w_1 e^-$ for some $e \in E$. Otherwise we would have $w = w_1 e_m^+$; hence, $f(e_m) \neq n$ and $f(e_m) = 1$, since we are in case (b); but this is case (b1).

Furthermore, there is no $e_1 \in E$ with $f(e_1) = n - 1$. Otherwise, we would find a value k of f among the $n > |E|$ possible values such that $1 < k < n$ and $k - 1$ is not a value of f (note that $f(e) = n$); hence, we could apply case (a).

Now proceed as in case (a), except that w_1' is defined by $w' = w_1'(e')^-$; in particular, $v = v_1(l(e)_n, e)$. At the end, we can conclude that $e = e_2$ since $n = f_2(e_2) = f_1(e_2) + 1$ and only e has f_1-value $n - 1$. Hence, $(w, l, f) = (w_2, l, f_2) \in swap^*(w', l, f')$.

EXAMPLE. Let us construct a v according to the proof of 6.7 for two examples with $n = 4$. First, let (w, l, f) be given by $w = 1^+2^+3^+1^-2^-$, $l(\{1, 2, 3\}) = \{a\}$, $f(\{1, 2\}) = \{4\}$ and $f(3) = 1$. Cases (a) and (b1) above are not applicable; (b2) shows that v must end $(a_4, 2)$. For the prefix (w_1, f_1, l_1) we have $f_1(2) = 3$ and can apply case (a); we get that $(a_4, 2)$ must be preceded by $(a_3, 2)$. Then we must apply (b2) again. We finally arrive at the sequence $v = (a_1, 1)$ $(a_2, 1)(a_1, 2)(a_3, 1)(a_2, 2)(a_1, 3)(a_4, 1)(a_3, 2)(a_4, 2)$. If we observe $abs(v)$, we see an a starting and proceeding to phase 2 before a second a starts. Hence, the first a_3 belongs to the first a-action, the succeeding a_2 to the second one. After the third a-action has started some a finishes, and this must be the first a since only this one is in its third phase.

As a second example, consider (w', l, f) with $w' = 1^+2^+3^+2^-1^-$ and with l and f as above. In this case, the above proof yields $v = (a_1, 1)(a_1, 2)(a_2, 2)(a_2, 1)(a_1, 3)$ $(a_3, 2)(a_4, 2)(a_3, 1)(a_4, 1)$ or $v = (a_1, 1)(a_2, 1)(a_1, 2)(a_2, 2)$ $(a_1, 3)(a_3, 2)(a_4, 2)(a_3, 1)(a_4, 1)$. In these sequences, one can see that the third a never gets to its second phase; thus, the a_3's and a_4's are phases of the first or second a. But the first and the second a get to the same phase when $(a_1, 2)$ or $(a_2, 2)$ occurs. Thus, when the first a_4 occurs in $abs(v)$, we can not be sure whether it belongs to the first or the second a. Hence, the structure of starts and ends corresponding to $abs(v)$ is w' or w as above; this fits our result, since $(w', l, f) \in swap(w, l, f)$.

It might be a little surprising that the first and second a get confused in their first or second phase already. Constructing a suitable v "by hand," we would possibly end up with $(a_1, 1)(a_2, 1)(a_1, 2)(a_3, 1)(a_2, 2)(a_1, 3)(a_3, 2)(a_4, 2)(a_4, 1)$. Here, we keep the three a-actions in different phases as long as possible. Only when we have to finish the second a after the start of the third a, we have to move it to the same phase as the first a, which is in this case the third phase. The reason that the proof gives v as described above is, that the construction works inductively; i.e., v is in a way constructed from back to front. Seen from the back, it is clear that the second a overtakes the first one, hence they must get to the same phase at some stage anyway.

Now we are ready to prove the result of this section.

THEOREM 6.8. *If nets are $split_n$-equivalent for all $n \in \mathbb{N}$, then they are swap-equivalent.*

Proof. Consider nets N_1 and N_2 that are $split_n$-equivalent for all $n \in \mathbb{N}$ and some $(w, l) \in swap\text{-}IW(N_1)$. Let (w, l) be defined over E and choose some $n \geqslant 2$ with $n > |E|$. Choose some n-phase interval sequence (w, l, f) and according to Lemma 6.7 a concrete n-refinement v. By Lemma 6.6, $abs(v)$ is an abstract n-refinement of (w, l) and by Lemma 5.7 in $L(split_n(N_1)) = L(split_n(N_2))$. Again by Lemma 5.7, we find some $(w', l') \in swap\text{-}IW(N_2)$ such that $abs(v)$ is an abstract n-refinement of (w', l'). By 6.6, $abs(v)$ is an abstract n-refinement of some (w', l', f').

Lemma 6.7 shows $(w, l, f) \in swap^*(w', l', f')$, Lemma 6.3 gives $(w, l) \in swap^*(w', l')$. Hence, we have $(w, l) \in swap\text{-}IW(N_2)$. We conclude by symmetry that $swap\text{-}IW(N_1) = swap\text{-}IW(N_2)$.

7. FURTHER CONSEQUENCES

In this section, we will sketch two consequences of our main result. The first concerns parallel composition with synchronization as in TCSP: in $N \parallel_A N'$, where $A \subseteq \Sigma$, the systems N and N' work in parallel; in general, they perform actions independently, but actions from A they have to perform together. The net $N \parallel_A N'$ is constructed from N and N' by merging, for each $a \in A$, each a-labelled transition of N with each a-labelled transition of N'; i.e., these transitions are replaced by all possible pairs. See [Vog92] for a formal definition.

We will see below that swap-equivalence is not a congruence for this important operation for the modular construction of systems. Thus, swap-equivalence is not adequate for comparing components of systems, and neither for coarse system descriptions from which final systems should be constructed hierarchically by action refinement. It *is* adequate for the comparison of completely specified systems, where we are interested in the duration of system runs, but do not know how long the single actions will take in actual implementations.

If we are looking for a linear-time semantics that supports action refinement and parallel composition, we should take interval semiwords, which give a congruence for both families of operations [Vog92, Theorems 5.3.18 and 5.3.19]. As a consequence of our main result, we will see in the following that interval-semiword equivalence is already fully abstract for language equivalence, parallel composition, and a very simple form of action refinement, namely splitting.

It is quite easy to turn an action-labelled partial order p into a net $N(p)$ such that its semiwords are p, its prefixes and their augmentations. E.g., for p in Fig. 2 we can choose $N(p) = N$ in Fig. 1; compare also [Vog92, pp. 186, 187]. If p is an action labelled interval order, then for each net N, $ISW(N \parallel_\Sigma N(p))$ contains only prefixes of p and their augmentations, i.e., $ISW(N \parallel_\Sigma N(p)) \subseteq ISW(N(p))$, and it contains p if and only if $p \in ISW(N)$. See [Vog92, Definition 3.1.5 and Theorem 5.3.19]. Since swapping preserves the sizes of the event set and of the ordering relation and since p is unique with these sizes in $ISW(N(p))$, we conclude from $ISW(N \parallel_\Sigma N(p)) \subseteq ISW(N(p))$ that $p \in swap\text{-}ISW(N \parallel_\Sigma N(p))$ if and only if $p \in ISW(N \parallel_\Sigma N(p))$ if and only if $p \in ISW(N)$.

Now consider N and N' in Fig. 1 and q in Fig. 2. N and N' are not interval-semiword equivalent, since $q \in ISW(N')$ and $q \notin ISW(N)$. This implies, by the above, that $N \parallel_\Sigma N(q)$ and $N' \parallel_\Sigma N(q)$ are not swap-equivalent, i.e., swap-equivalence is not a congruence for parallel composition.

The same argument works in general: for any two nets N and N' that are not interval-semiword equivalent, there is a context $split_n(. \parallel_\Sigma N(p))$ in which N and N' give different languages. Hence:

THEOREM 7.1. *Interval-semiword equivalence is fully abstract for language equivalence, parallel composition, and splitting, $split_2$ resp.*

As a second consequence of Theorem 5.9(i), we will sketch how the validity of certain equations in formal language theory can be checked. The expressions in these equations are built from variables with the operations concatenation $\cdot$, choice or union $+$ and shuffle $\parallel$. We call an equation for two such expressions *valid*, if it is true for all substitutions of words for the variables. It is easy to see that it is sufficient to check those substitutions where all the words have equal length and each letter appears at most once; such substitutions are essentially splittings. The other substitutions can be obtained by homomorphisms that rename letters to letters or to the empty word λ.

For each expression P we can construct a net $N(P)$ such that for all n the language described by P under the substitution $split_n$ is just—more or less—$L(split_n(N(P)))$. A thorough treatment is beyond the scope of this paper; we just show example nets $N(P)$ below. For a formal treatment, we would have to modify the definitions of $L(N)$, $ISW(N)$,

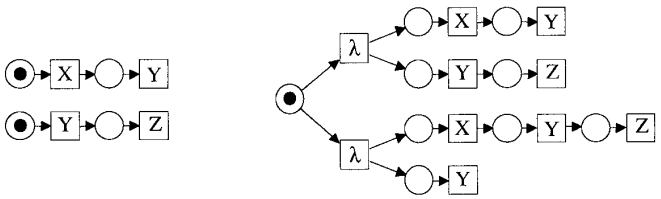


FIGURE 14

and *swap-ISW*(N) such that only "complete" runs of N would be considered; these modified definitions could be related to our definitions by introducing a special action $\surd$ to signal completion. But we leave out the details here and just say that Theorems 5.9, 5.10 and 5.11 can be modified accordingly—while 5.12 obviously cannot. From this we get that $P = Q$ is valid if and only if $N(P)$ and $N(Q)$ are swap-equivalent.

As an example, we consider the equation $XY \parallel YZ = (XY \parallel YZ) + (XYZ \parallel Y)$. Figure 14 shows the nets for the two expressions; observe that $\parallel$ corresponds to $\parallel_\varnothing$ for nets.

These nets are in fact very similar to our standard pair of nets from the introduction. To deduce swap-equivalence we simply have to observe that the interval semiword in Fig. 15 belongs to *swap-ISW*($N(XY \parallel YZ)$). Hence, the equation is true for all words X, Y and Z.

Note that the equation fails for languages; by the result mentioned in the introduction, it must fail since the nets in Fig. 14 do not have the same interval semiwords due to the one shown in Fig. 15. Let $X = \{a\}$, $Y = \{b_1 b_2, c_1 c_2\}$ and $Z = \{d\}$; then only the language belonging to the right-hand side contains $b_1 a c_1 c_2 d b_2$.

8. CONCLUSION

We have defined the swap-interval-semiword semantics of Petri nets and have shown that this partial order semantics is fully abstract with respect to language equivalence and splitting, which is a particular case of action refinement. This new semantics makes fewer distinctions than interval-semiword semantics, which is fully abstract with respect to language equivalence and general action refinement; i.e., the new semantics does not induce a congruence for general action refinement.

This result has to be contrasted with the following result on the level of bisimulations. ST-bisimulation, which corresponds to the interval idea on the level of bisimulations, is not only fully abstract with respect to bisimulation and general action refinement, but also with respect to bisimulation and splitting.

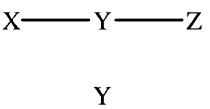


FIGURE 15

A natural question is what the situation is like for other semantics in the linear time-branching time spectrum. For example, I conjecture that failure semantics can be treated similarly to the treatment of language presented in this paper; hence, I conjecture that on the level of failure semantics full abstractness for splitting is different from full abtractness for general action refinement; see [Vog92] for the latter.

More challenging is the question which ingredient of bisimulation makes bisimulation different from the language in this comparison of splitting and action refinement. Is this ingredient characterisic for bisimulation? More precisely, is bisimulation the coarsest semantics in the linear time–branching time spectrum such that full abstractness for splitting and for general action refinement coincide?

We have also placed swap-interval-semiword equivalence in the interleaving–"true concurrency" spectrum, and have seen how it can be checked for small examples using some nice closure properties of the new semantics. It seems that the new equivalence is decidable for finite nets, but no decision algorithm is known so far. Finally, we have sketched how our full abstraction result for splitting can be applied to certain equations in formal language theory and to prove that interval-semiword equivalence is fully abstract for language equivalence, parallel composition and splitting.

APPENDIX

In Theorem 2.7, we have studied the representation of finite interval orders. One could also be interested in infinite system runs and, thus, in representing countably infinite interval orders by infinite sequences; this is shortly discussed in this appendix.

In fact, all definitions before Theorem 2.7 and 2.1 and 2.6 also work for the countably infinite case [Fis85]. In this case, 2.6 still shows that a $co^{\pm}$-equivalence class consists of starts only or of ends only, but now we can only say that every two start classes are separated by an end class with respect to $<^{\pm}$ and vice versa; furthermore, each start class is succeeded by an end class and each end class is preceded by a start class due to 2.6(iii). As regards 2.7(i), we have to be careful since in the countable case not every total order, hence not every linearization, corresponds to a sequence.

Let some countably infinite interval order $p = (E, <)$ be given. If there are infinitely many e_1 with $e_1 < e$ or $e_1\ co\ e$ for some given $e \in E$, then for each of these e_1 we have $e_1^+ <^{\pm} e^-$, hence in a closed representation we would have infinitely many e_1^+ before e^-, which is impossible.

If this case does not apply, p is called *initially finite* in [JK93]. For initially finite p, the definition of $<^{\pm}$ shows that each end e^- has only finitely many starts as predecessors. Since each end is preceded by the corresponding start in $(E^{\pm}, <^{\pm})$, this shows that e^- has only finitely many ends as predecessors. Furthermore, if $e^-\ co^{\pm}\ e_1^-$ then

$e_1^+ <^\pm e^-$; hence, the $co^\pm$-equivalence class of e^- is finite. Similarly, e^+ has only finitely many predecessors and finitely many concurrent (w.r.t. $<^\pm$) starts, since all of these are predecessors of e^-. Thus, in a linearization w of $(E^\pm, <^\pm)$, each element has only finitely many predecessors and w is a sequence.

So we get that a countable interval order has a closed representation if and only if it is initially finite. This result is essentially given in [JK93], since a closed representation more or less coincides with what is in [JK93] called an injective discrete interval representation with positive values. Furthermore, the closed representations of an initially finite interval order $(E, <)$ are exactly the linearizations of $(E^\pm, <^\pm)$. An analogue of 2.7(ii) holds for a suitable definition of commuting (possibly infinitely often) elements in an infinite sequence.

If we also consider arbitrary representations, we first observe the following: If $(E, <)$ is countable and initially finite, then each $e \in E$ has a successor, hence e^- must occur in a representation. Thus, each representation is closed.

Now let $(E, <)$ be a countable interval order that is not initially finite—and hence no closed representation exists. If for some $e_1 \in E$ there are infinitely many e for which we can find some e' with $e \; co \; e'$ and $e' < e_1$, then e_1^+ has infinitely many predecessors in $(E^\pm, <^\pm)$, hence no representation exists. If this is not the case, we call $(E, <)$ *weakly initially finite*. Figure 16 shows two examples, which are not initially finite.

Observe that for such an $(E, <)$ each event has only finitely many predecessors (since co is reflexive). Hence:

(★) Each start has only finitely many predecessors in $(E^\pm, <^\pm)$.

Since $(E, <)$ is not initially finite, but weakly initially finite, it must have some e and infinitely many events concurrent to it. In $(E^\pm, <^\pm)$, e^- has the starts of all these events as predecessors and so have all the ends in the $co^\pm$-equivalence class C of e^-. Hence, a representation of $(E, <)$ cannot contain an element of C. By (★), C is not followed by any element in $(E^\pm, <^\pm)$. But all other ends must be followed by some start, again by 2.6(v). We conclude that each representation contains exactly the elements of $E^\pm - C$.

Furthermore, we conclude that the $co^\pm$-equivalence classes form an alternating sequence of start- and end-classes, beginning with a start class, and this sequence is followed by C, which consists of all maximal elements of $(E^\pm, <^\pm)$. All the $co^\pm$-equivalence classes $\neq C$ consisting of ends are separated by a start from C; from (★) we see that they are finite and that there elements have finitely many predecessors.

If all $co^\pm$-equivalence classes—except possibly C—are finite, then each linearization gives a sequence on $E^\pm - C$. Thus, a representation exists and all other representations can be obtained by commuting ends and commuting starts.

If some $co^\pm$-equivalence class $C_1 \neq C$ is infinite, it consists of starts; the alternating sequence of $co^\pm$-equivalence classes is finite and ends with $C_1 C$. Each linearization of $(E^\pm, <^\pm)$ that is a sequence on $E^\pm - C$ consists of a finite sequence on $E^\pm - C_1 - C$ followed by the elements of C_1 in some arbitrary sequence (and followed by the elements of C). Again we see that a representation exists and all other representations can be obtained by commuting ends and commuting starts.

Collecting all our conclusions we obtain:

> An infinite interval order p has a representation if and only if it is weakly initially finite. All representations can be obtained from a given one by commuting ends and commuting starts. The representations are closed if and only if p is initially finite.

FIGURE 16

Received May 4, 1995; final manuscript received March 5, 1996

REFERENCES

[AH93] Aceto, L., and Hennessy, M. (1993), Towards action-refinement in process algebras, *Inform. and Comput.* **103**, 204–269.

[AM96] Aceto, L., and Murphy, D. (1996), On the ill-timed but well-caused, *Acta Informat.* **33**, to appear.

[BDKP91] Best, E., Devillers, R., Kiehn, A., and Pomello, L. (1991), Concurrent bisimulations in Petri nets, *Acta Informat.* **28**, 231–264.

[BF88] Best, E., and Fernandéz, C. (1988), "Nonsequential Processes. A Petri Net View," EATCS Monographs on Theoretical Computer Science, Vol. 13, Springer-Verlag, Berlin/New York.

[Dev92] Devillers, R. (1992), Maximality preservation and the ST-idea for action refinement, *in* "Advances in Petri Nets 1992" (G. Rozenberg, Ed.), Lecture Notes in Computer Science, Vol. 609, pp. 108–151, Springer-Verlag, Berlin/New York.

[DG95] Degano, P., and Gorrieri, R. (1995), A causal operational definition of action refinement, *Informat. and Comput.* **122**, 97–191.

[Die90] Diekert, V. (1990), "Combinatorics on Traces," Lecture Notes in Computer Science, Vol. 454. Springer-Verlag, Berlin/New York.

[Fis85] Fishburn, P. C. (1985), "Interval Orders and Interval Graphs," Wiley, New York.

[GG89a] Glabbeek, R. J. v., and Goltz, U. (1989), Partial order semantics for refinement of actions—Neither necessary nor always sufficient, but appropriate when used with care, *EATCS Bull.* **38**, 154–163.

[GG89b] Glabbeek, R. J. v., and Goltz, U. (1989), Equivalence notions for concurrent systems and refinement of actions, *in* "MFCS 89" (A. Kreczmar, and G. Mirkowska, Eds.), Lecture Notes in Computer Science, Vol. 379, pp. 237–248, Springer-Verlag, Berlin/New York.

[GL95] Gorrieri, R., and Laneve, C. (1995), Split and ST bisimulation semantics, *Inform. and Comput*. **118**, 272–288.

[Gla90] Glabbeek, R. J. v. (1990), The refinement theorem for ST-bisimulation semantics, *in* "Programming Concepts and Methods, Proceedings, IFIP Working Conference," (M. Broy and C. B. Jones, Eds.), pp. 27–52, Elsevier, (North-Holland), Amsterdam.

[GR83] Goltz, U., and Reisig, W. (1983), The non-sequential behaviour of Petri nets, *Inform. and Control* **57**, 125–147.

[Gra81] Grabowski, J. (1981), On partial languages, *Fund. Informat*. **4**, No. 2, 428–498.

[GV95] Glabbeek, R. J. v., and Vaandrager, F. (1995), "The difference between Splitting in n and $n+1$," Technical Report CS-R9553, CWI.

[Hen88] Hennessy, M. (1988), Axiomatising finite concurrent processes, *SIAM J. Comput*. **17**, 997–1017.

[JK93] Janicki, R., and Koutny, M. (1993), "Representations of Discrete Interval Orders and Semi-orders," Technical Report 93-02, Dept. Comp. Sci. Sys., McMaster University, Hamilton, Ontario.

[JM92] Jategaonkar, L. A., and Meyer, A. R. (1982), Testing equivalence for Petri nets with action refinement, *in* "CONCUR '92" (W. R. Cleaveland, Ed.), Lecture Notes in Computer Science, Vol. 630, pp. 17–31, Springer-Verlag, Berlin/New York.

[Kie88] Kiehn, A. (1988), On the interrelationship between synchronized and non-synchronized behaviour of Petri nets, *J. Inf. Process. Cybernet. EIK* **24**, 3–18.

[Lar88] Larsen, K. S. (1988), "A Fully Abstract Model for a Process Algebra with Refinement," Master's thesis, Dept. Comp. Sci., Aarhus University.

[Mey95] Meyer, A. R. (1995), Concurrent process equivalences: Some decision problems (invited talk), *in* "STACS 95" (E. Mayr and C. Puech, Eds.), Lecture Notes in Computer Science, Vol. 900, p. 349, Springer-Verlag, Berlin/New York.

[NEL89] Nielsen, M., Engberg, U, and Larsen, K. (1989), Partial order semantics for concurrency, *in* "Proc. REX School / Workshop Linear Time, Branching Time and Partial Order in Logic and Models of Concurrency, Noordwijkerhout, 1988" (J. W. Bakker *et al.*, Eds.), Lecture Notes in Computer Science, Vol. 354, pp. 523–548, Springer-Verlag, Berlin/New York.

[Pet81] Peterson, J. L. (1981), "Petri Net Theory," Prentice–Hall, New York.

[Rei85] Reisig, W. (1985), "Petri Nets," EATCS Monographs on Theoretical Computer Science, Vol. 4, Springer-Verlag, Berlin/New York.

[Ren93] Rensink, A. (1993), "Models and Methods for Action Refinement," Ph.D. thesis, Faculteit der Informatica, Universiteit Twente.

[Sta81] Starke, P. H. (1981), Processes in Petri nets, *J. Inform. Process. Cybernet. EIK* **17**, 389–416.

[Vog91] Vogler, W. (1991), Failures semantics based on interval semiwords is a congruence for refinement, *Distrib. Comput*. **4**, 139–162.

[Vog92] Vogler, W. (1992), "Modular Construction and Partial Order Semantics of Petri Nets," Lecture Notes in Computer Science, Vol. 625, Springer-Verlag, Berlin/New York.

[Vog93] Vogler, W. (1993), Bisimulation and action refinement, *Theoret. Comput Sci*. **114**, 173–200.

[Vog95] Vogler, W. (1995), Timed testing of concurrent systems, *Inform. and Comput*. **121**, 149–171.

Printed in Belgium