In Proceedings of CONCUR 95, LNCS n. 962

Characterizing Behavioural Congruences for Petri Nets

Mogens Nielsen[*] *Lutz Priese*[**] *Vladimiro Sassone*[*,◇]

[*]**BRICS** – University of Aarhus, [**]University of Koblenz

Abstract. We exploit a notion of *interface* for Petri nets in order to design a set of net *combinators*. For such a calculus of nets, we focus on the *behavioural congruences* arising from four simple notions of behaviour, *viz.*, traces, maximal traces, step, and maximal step traces, and from the corresponding four notions of bisimulation, *viz.*, weak and weak step bisimulation and their maximal versions. We characterize such congruences via *universal contexts* and via *games*, providing in such a way an understanding of their discerning powers.

Introduction

In the early days of Petri net theory some important classes of Petri nets (**PN**) were introduced, such as Marked Graphs (**MG**), State Machines (**SM**), Free Choice nets (**FC**), Simple nets (**SN**), and inhibitory Petri nets (**iPN**). The 'expressive power' hierarchy among these classes is folklore, namely

$$\mathbf{MG}, \mathbf{SM} \prec \mathbf{FC} \prec \mathbf{SN} \prec \mathbf{PN} \prec \mathbf{iPN},$$

where, for instance, $\mathbf{SN} \prec \mathbf{PN}$ reads that simple nets do not possess the modelling power of arbitrary Petri nets. In proving this fact, Patil [18] shows that no simple net possesses the behaviour of the so-called '3-smoker-net', i.e., that for a certain notion $\approx$ of behavioural equivalence, '3-smoker-net' $\not\approx SN$, for all simple nets *SN*. However, $\approx$ was not defined formally; the clever and convincing argument remained at a rather informal level. The same consideration applies to various decomposition results of that time. For instance, another folklore result at the Project MAC of the MIT was that every Petri net can be obtained as 'composition' of few, very simple components. Also in this case, the semantic and the composition operations were not defined formally. The proofs simply gave a decomposition technique preserving firing sequences and some 'concurrency properties'. Looking back to these and to related works (e.g. [6, 4, 24, 10, 19]), it is now clear that the intuition behind $\approx$ was the today-well-know idea of behavioural congruence, i.e., for a fixed notion of behaviour $\mathcal{B}$,

$$N_0 \approx N_1 \quad \Leftrightarrow \quad \mathcal{B}\big(\mathcal{C}[N_0]\big) = \mathcal{B}\big(\mathcal{C}[N_1]\big), \text{ for all 'net contexts' } \mathcal{C}.$$

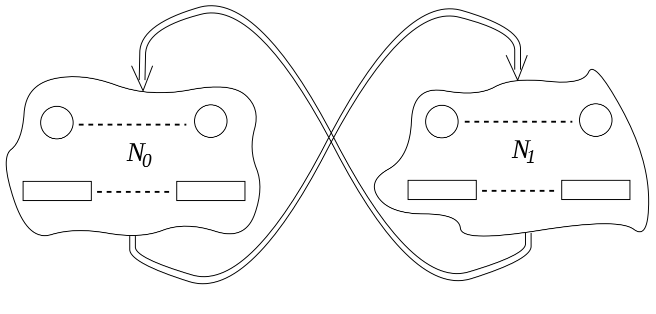
Inspired by these ideas, in this paper we plan to formalize the notions of 'composition' and, consequently, of 'net context', drawing on the experience of developments in concurrency theory. Moreover, we focus on some simple notions of behaviours, namely four kinds of traces for labelled nets with (invisible) τ-transitions, and on the corresponding branching behavioural equivalences, namely four kinds of (weak) bisimulation, studying the behavioural congruences which arise from them, and providing characterizations of these congruences.

* **B**asic **R**esearch **in** **C**omputer **S**cience, Centre of the Danish National Research Foundation.
◇ Supported by EU **H**uman **C**apital and **M**obility grant ERBCHBGCT920005.

Our first aim, therefore, is to define a set **CM** of combinators of Petri nets. Several works have focused on algebraic aspects of Petri nets, e.g. [16, 23, 11, 5, 7, 3, 2, 14, 21]. Here we shall consider a minimal set of combinators, focusing on (versions of) parallel composition, relabelling, restriction, and recursion, and allowing a rather general form of interaction.

Similarly to [2], our approach is entirely based on a notion of *interface* for Petri nets, introduced in Section 1. In fact, although some of the operations of composition we have in mind could be safely described on ordinary nets, the most interesting ones, those entailing 'cooperation', give sensible results only when defined through a notion of interface. Informally, an interface for a net N is a selection of places and transitions of N which specifies what parts of N are *public*, i.e., accessible to the environment, and what parts are *private* to N. The private places and transitions cannot be accessed and, therefore, they cannot be used for connecting N with other nets. More precisely, net interfaces are built out of two components: an 'input' interface, consisting of places, and an 'output' interface, consisting of transitions. Intuitively, the input interface provides the buffers in which the tokens arriving from the environment are gathered, whilst the output interface sends tokens out to the environment. From a different viewpoint, interfaces can be viewed as a simple form of *typing* of nets on which the combinators in **CM** can be built.

The partition of interfaces in input and output parts is the actual key to the design of a significant, yet manageable, set of combinators for nets. In fact, the main way of combining nets provided by **CM** is directly suggested by such a notion: it consists of connecting the outputs of one net to the inputs of another net and, possibly, vice versa, as schematically shown by the following picture.



Following the principle of considering as simple operations as possible, we shall realize this by means of two more basic combinators: *par*, which simply puts its two arguments side by side, and *add*, which augments its argument by a new arc from an interface-transition to an interface-place. The operation illustrated above is then obtained by repeatedly applying *add* to $par(N_0, N_1)$. Clearly, *add* in isolation provides an interesting form of *recursion* consisting of feeding back outputs to inputs. We moreover introduce combinators dealing with *relabelling* and *hiding* intended to make easier the description of (large) modular systems. It is important to notice that, respecting our intuition about input and output parts of interfaces, it is *not* possible to connect inputs to outputs, i.e., to inspect other net's inputs or to feed other net's outputs. The only possible kind of cooperation is realized by *sending* and *receiving* tokens. Besides matching the current ideas in concurrency theory, our main observation is that this restriction makes **CM** manageable: it is difficult for us to imagine formal results along the lines of this paper without it. Notice that, enriching our setting with the operation of adding arcs from inputs to outputs, *every* finite net could be built in the calculus from the 'single-place' net and the 'single-transition' net. However, our choice of combinators still seems to provide a rather expressive 'calculus of nets'. We provide some evidence of this in Section 2, where we present a derived combinator *plus* modelling an interesting form of nondeterministic composition similar to the *internal choice* of process algebras [9, 8], and we express the *task scheduler* of [12, 13] in our calculus.

In Section 3, we focus on four simple notions of behaviour $\mathcal{B}$, namely traces and maximal traces (the *interleaving* case), and step traces and maximal step traces (the *noninterleaving* one). For each of these notions, we consider the corresponding behavioural equivalence $\approx_{\mathcal{B}}$ and the corresponding bisimulation $\leftrightarrows_{\mathcal{B}}$. Then, we focus on the largest congruences for **CM** contained in $\approx_{\mathcal{B}}$ and $\leftrightarrows_{\mathcal{B}}$, written, respectively, as $\approx^c_{\mathcal{B}}$ and $\leftrightarrows^c_{\mathcal{B}}$.

Section 4 and Section 5 are devoted to simple characterizations of the behavioural congruences $\approx^c_{\mathcal{B}}$ and $\leftrightarrows^c_{\mathcal{B}}$, which, from the technical viewpoint, are the main results of the paper. In Section 4, we identify a minimal set of contexts which is *universal* for $\approx^c_{\mathcal{B}}$. More precisely, for each pair N_0 and N_1 of nets with interface there exists a readily-identified context $\mathcal{C}$ such that N_0 and N_1 are $\approx_{\mathcal{B}}$-congruent if and only if $\mathcal{C}$ does not distinguish them. This result lifts to $\leftrightarrows^c_{\mathcal{B}}$ provided a rather mild and reasonable condition is imposed on nets with interface, namely that interface-transitions must carry visible labels.

The nature of these results allows to transport results from the equivalences we considered to the corresponding congruences. For instance, we obtain fully abstract models for $\approx^c_{\mathcal{B}}$ in terms of formal languages; these results, reported in the full version of the paper [17], are omitted in this exposition. As a further example, following recent accounts of bisimulation in terms of games, e.g. [22, 15], we present in Section 5 game theoretic characterizations of $\approx^c_{\mathcal{B}}$ and $\leftrightarrows^c_{\mathcal{B}}$. More precisely, we design for each of our $\mathcal{B}$'s a two-player game played on nets with interface N_0 and N_1 in such a way that a designated player has a winning strategy if and only if N_0 and N_1 are $\approx_{\mathcal{B}}$-congruent. A corresponding result, modulo the condition on labels mentioned above, is obtained for $\leftrightarrows^c_{\mathcal{B}}$.

1 Petri Nets, Interfaces, and Combinators

In this section we recall the definition of Petri nets (see also [20]), we introduce a notion of interface for nets, and design a small set of combinators by means of which nets can be composed to and interact with each other via interfaces.

Remark. A *multiset* on a set P is a function $\mu\colon P \to \mathbb{N}$; the *union* of multisets μ_0 and μ_1 on P is the multiset $\mu_0 + \mu_1$ such that $(\mu_0 + \mu_1)(p) = \mu_0(p) + \mu_1(p)$. We shall use $\mu(P)$ to denote the set of multisets on P. We make the convention that, whenever we consider words of multisets, we identify the empty word ϵ with the empty multiset, i.e., the function yielding 0 on all $p \in P$.

A *Petri net* can be regarded as an automaton whose states are represented by distributions of 'tokens' in a set of atomic state components called *places*. Similarly, the transition of state are determined by the concurrent 'firing' of multisets of atomic computational steps called *transitions*. Here we are interested in finite nets whose transitions are labelled by (possibly invisible) actions. To this aim, we shall use a *countable* set *Act* of *visible* actions $\alpha_1, \alpha_2, \alpha_3, \ldots$, and a distinguished *invisible* action τ.

Definition 1.1 (*Labelled Petri Nets*)
A (finite) labelled Petri net is a tuple $N = (T_N, P_N, F_N, s_N, \lambda_N)$, where

- ▷ *T_N is a finite set of transitions, P_N is a finite set of places, and $T_N \cap P_N = \varnothing$,*
- ▷ *$F_N \colon (T_N \times P_N) \cup (P_N \times T_N) \to \mathbb{N}$ is the flow (multi)relation defining directed (multi)arcs between transitions and places,*
- ▷ *$s_N \in \mu(P_N)$ is the initial state (or marking),*
- ▷ *$\lambda_N \colon T_N \to Act \cup \{\tau\}$ is a labelling function.*

We shall use **PN** *to refer to the class of (finite) labelled Petri nets.*

The dynamic of nets is regulated by the notion of *fireable step*. Intuitively, a finite multiset of transitions, in the following called *step*, may fire at given state if the latter provides enough resources.

Definition 1.2 (*Steps and Step Sequences*)
A step $X \in \mu(T_N)$ is fireable at state $s \in \mu(P_N)$, if $\sum_{t\in T_N} X(t)\cdot F_N(p,t) \le s(p)$ for all $p \in P_N$; the firing of X at s leads to the state s', denoted as $s[X\rangle s'$, where $s'(p) = s(p) + \sum_{t\in T_N} X(t)\cdot(F_N(t,p) - F_N(p,t))$, for all $p \in P_N$.

A step sequence of N is a sequence $s[X_1\rangle s_1 \cdots s_{n-1}[X_n\rangle s_n$ of firings of steps. A step sequence $s[X_1 \cdots X_n\rangle s_n$ is maximal if no non-empty step is fireable at s_n.

We shall write respectively $S(N)$ and $S_m(N)$ for the sets of step sequences and maximal step sequences of N fireable at the initial state s_N.

Considering the sequences of multisets of visible actions labelling the transitions occurring in step sequences and maximal step sequences, we obtain the following classical notions of languages (sets of *traces*) of Petri nets. When needed, we shall single out sequences corresponding to maximal step sequences by marking them with a distinguished symbol ✓. This allows to represent faithfully enough 'non-termination', yet avoiding to deal explicitly with infinite sequences.

Definition 1.3 (*Step Languages*)
For X a step of N, let $\hat\lambda_N(X)$ be the multiset of non-τ actions of X, i.e., for all $\alpha \in Act$, $\hat\lambda_N(X)(\alpha) = \sum_{t\in\lambda_N^{-1}(\alpha)} X(t)$. For a step sequence $SX = s[X_1 \cdots X_n\rangle s_n$ of N, let $\hat\lambda_N(SX)$ be $\hat\lambda_N(X_1 \cdots X_n) = \hat\lambda_N(X_1) \cdots \hat\lambda_N(X_n)$. Then, the step language of N is the set $\mathcal{S}(N) = \big\{\hat\lambda_N(SX) \bigm| SX \in S(N)\big\} \subset \mu(Act)^*$, and, writing SL for $\big\{\hat\lambda_N(SX) \bigm| SX \in S_m(N)\big\}$, the maximal-step language of N is the set $\mathcal{S}_m(N) = \big\{\mu_1 \cdots \mu_n\checkmark \bigm| \mu_1 \cdots \mu_n \in SL\big\} \cup \mathcal{S}(N) \subset \mu(Act)^* \cdot \{\epsilon, \checkmark\}$.

The step sequences whose steps consist of *at most* one transition are called *firing sequences.* We denote respectively by $\mathcal{L}(N)$ and $\mathcal{L}_m(N)$ the subsets of $\mathcal{S}(N)$ and $\mathcal{S}_m(N)$ corresponding to firing sequences. The sets $\mathcal{L}(N)$ and $\mathcal{L}_m(N)$ are called, respectively, the *language* and the *maximal-trace language* of N.

We aim at defining a minimal set of net combinators which will be expressive enough to form a rudimentary calculus of nets. It should certainly include operations allowing (forms of) communication, parallel composition, relabelling, restriction, and recursion. Pondering the issue, it becomes soon evident that, to avoid a chaotic 'structural' calculus where everything is permitted, some restrictions on the allowed connections via places and transitions must be imposed. Our solution is interfaces and their input/output partitions. An interface is an ordered collection of places, the 'input', and an ordered collection of transitions, the 'output', which specifies the parts of the net that are public and can, therefore, be used by other nets to interact. Interfaces readily suggest a reasonable discipline of interaction: connections between nets should go from outputs to inputs, involving only public components. This formalizes the well-motivated and solid intuition that the only allowed interactions are achieved by *sending* and *receiving* along interfaces, to be thought of as communication channels.

Definition 1.4 (*Labelled Nets with Interface*)
A net with interface is a structure $p_1,\ldots,p_n;t_1,\ldots,t_m \rhd N$, *where* N *is a net in* **PN**, *and* $p_1,\ldots,p_n \in P_N$, $t_1,\ldots,t_m \in T_N$ *are all distinct.*

We shall use **IPN** *to denote the class of nets with interface.*

Stressing the intuition of interface places and transitions as bound variables, we consider net with interface up to renaming. More precisely,

$$(p_1,\ldots,p_n;t_1,\ldots,t_m \rhd N) \equiv (p'_1,\ldots,p'_n;t'_1,\ldots,t'_m \rhd N')$$

if there is an isomorphism between N and N' which maps p_i to p'_i and t_j to t'_j, for $1 \leq i \leq n$ and $1 \leq j \leq m$. We shall often use $\vec{p}$ as a shorthand for $p_1,\ldots,p_n$; in such a case, $|\vec{p}|$ stands for n. Analogously for $\vec{t}$.

Definition 1.5 (*Combinators of Nets with Interface*)
The set **CM** *of combinators of nets with interface consists of the combinators defined by the following rules.*

- $\rhd$ $\dfrac{\vec{p}_0;\vec{t}_0 \rhd N_0 \quad \textit{and} \quad \vec{p}_1;\vec{t}_1 \rhd N_1 \quad \textit{disjoint}}{\textit{par}(\vec{p}_0;\vec{t}_0 \rhd N_0, \vec{p}_1;\vec{t}_1 \rhd N_1) = \vec{p}_0,\vec{p}_1;\vec{t}_0,\vec{t}_1 \rhd N_0 \| N_1}$

  where $N_0\|N_1$ *is the (componentwise) union of* N_0 *and* N_1*;*

- $\rhd$ $\dfrac{1 \leq i \leq |\vec{p}| \quad \textit{and} \quad 1 \leq j \leq |\vec{t}|}{\textit{add}(i,j,\vec{p};\vec{t} \rhd N) = \vec{p};\vec{t} \rhd N\langle p_i \hookleftarrow t_j\rangle}$

  where $N\langle p \hookleftarrow t\rangle$ *is the net* N *augmented with an arc from* t *to* p*;*

- $\rhd$ $\textit{rel}(\phi,\vec{p};\vec{t} \rhd N) = \vec{p};\vec{t} \rhd N[\phi]$,

  where $\phi\colon \textit{Act} \to \textit{Act} \cup \{\tau\}$ *is a 'relabelling' function, and* $N[\phi]$ *is obtained from* N *by relabelling via* ϕ *the transitions that carry non-*τ *actions;*

$$\triangleright \quad \frac{\max(P) \leq |\vec{p}| \quad and \quad \max(T) \leq |\vec{t}|}{hide(P, T, \vec{p}; \vec{t} \triangleright N) = \vec{p} \setminus P; \vec{t} \setminus T \triangleright N}$$

where P and T are finite sets of positive natural numbers ($\max(\varnothing) = 0$), and $\vec{x} \setminus X$ is the string obtained from $\vec{x}$ by removing x_i, for all $i \in X$.

Observe that, since the $par(\_,\_)$ combinator is defined explicitly only for disjoint nets, a renaming is generally needed before applying it to its arguments. This implies that no 'fusion' of nets is allowed by **CM**. Notice also that the operations dealing with interface places and transitions refer to them via the respective ordering, and they are not defined if the interfaces are not large enough. For instance, $add(i, j, \_)$ simply adds an arc from the ith place to the jth transitions of the interface. It provides both a form of recursion and, used in connection with $par(\_,\_)$, a form of 'asynchronous message passing' which feeds the inputs of a net with the outputs of another one. Finally, the self-explanatory $rel(\phi, \_)$ and $hide(P, T, \_)$ are intended to facilitate the description of modular systems.

Notation. For $\alpha(i)$ and $\beta(i)$ positive integer expressions in a positive integer variable i, we use $add_{i=1}^{k}(\alpha(i), \beta(i), \vec{p}; \vec{t} \triangleright N)$ for $add(\alpha(1), \beta(1), add(\cdots, add(\alpha(k), \beta(k), \vec{p}; \vec{t} \triangleright N) \cdots))$. Using the obvious associativity of $par(\_,\_)$, given $\vec{p}_1; \vec{t}_1 \triangleright N_1, \ldots, \vec{p}_k; \vec{t}_k \triangleright N_k$, we write $par(\vec{p}_1; \vec{t}_1 \triangleright N_1, \ldots, \vec{p}_k; \vec{t}_k \triangleright N_k)$, or also $par_{i=1}^{k}(\vec{p}_i; \vec{t}_i \triangleright N_i)$, for their parallel composition.

In the framework of computation theories, a context represents an environment in which a computing agent may operate. They are formalized as expressions with a hole $(\_)$ representing the place to be filled by the agent. In our case, **CM**-contexts are generated as follows, where i and j range over $\mathbb{N} \setminus \{0\}$, P and T over finite subsets of $\mathbb{N} \setminus \{0\}$, and ϕ over the functions $Act \to Act \cup \{\tau\}$.

$$\mathcal{C} ::= (\_) \mid \vec{p}; \vec{t} \triangleright N \mid par(\mathcal{C}, \mathcal{C}) \mid add(i, j, \mathcal{C}) \mid rel(\phi, \mathcal{C}) \mid hide(P, T, \mathcal{C}).$$

We shall write **CTX** to indicate the class of contexts $\mathcal{C}$.

The insertion of a net $\vec{p}; \vec{t} \triangleright N$ in a context $\mathcal{C}$ consists of replacing $(\_)$ by $\vec{p}; \vec{t} \triangleright N$ and evaluating the expression so obtained according to the rules in Definition 1.5. If the evaluation is not possible, because of the side conditions, we say that $\mathcal{C}[\vec{p}; \vec{t} \triangleright N]$ is undefined, in symbols $\mathcal{C}[\vec{p}; \vec{t} \triangleright N] = \bot$.

2 Two Examples: Nondeterministic Composition and Task Scheduler

In order to substantiate our claims on the expressiveness of our combinators, in this section we derive a combinator for a simple form of nondeterministic composition suitable for nets with interface; moreover, we consider Milner's example of a n-task scheduler, and we realize it using nets with interface and their combinators. As a byproduct, the section covers an interesting aspect of system composition, viz., nondeterminism, that we did not consider in **CM**. It may be observed that we allow ourselves to consider in our 'calculus' any net whatsoever. It is then noteworthy that the following constructions build on only two very simple nets, viz., *MCh* and *SCh*.

Driven by the notion of input interface, the following idea arises naturally: a sensible form of nondeterministic composition of nets $\vec{p}_0; \vec{t}_0 \triangleright N_0$ and $\vec{p}_1; \vec{t}_1 \triangleright N_1$

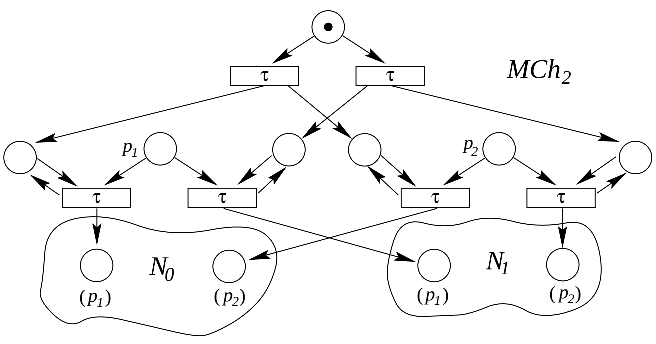
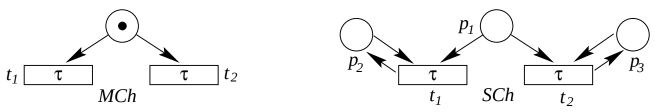


Figure A

with compatible input interfaces, i.e., $|\vec{p}_0| = n = |\vec{p}_1|$, is the net with n input channels which decides *nondeterministically*, once and for all, to forward the tokens it receives from the environment to N_0 or to N_1. The input interfaces of the component nets are private to the new net, while their outputs are still public. Of course, a symmetric combinator acting on the output interfaces can be conceived. For the time being, however, we limit ourselves to the inputs.

Let $\varnothing; t_1, t_2 \triangleright MCh$ and $p_1, p_2, p_3; t_1, t_2 \triangleright SCh$ be the nets shown below.



For $n \in \mathbb{N}$, let SCh_n be $par^n_{i=1}(p_1, p_2, p_3; t_1, t_2 \triangleright SCh)$ and consider

$$MCh_n = hide\Big(\{3i{-}1, 3i \mid i = 1, \ldots, n\}, \{1, 2\},\\ add^n_{i=1}\big(3i{-}1, 1, add^n_{i=1}(3i, 2, par(MCh, SCh_n))\big)\Big).$$

The transitions of SCh_n are initially *disabled* by looping places (see Figure A). The *MCh* component *nondeterministically* enables all the 'left' or all the 'right' branches of the *SCh*'s by feeding one token into the corresponding looping place. Since *MCh* cannot repeat its choice, this implements the 'once-and-for-all' strategy we wanted. Then, we can define $plus(\_, \_)$ in the following obvious way:

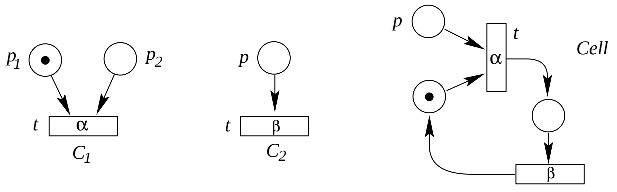
$$\frac{|\vec{p}_0| = n = |\vec{p}_1|}{plus(\vec{p}_0; \vec{t}_0 \triangleright N_0, \vec{p}_1; \vec{t}_1 \triangleright N_1) = \mathsf{A}_n(MCh_n, \vec{p}_0; \vec{t}_0 \triangleright N_0, \vec{p}_1; \vec{t}_1 \triangleright N_1)},$$

where A_n denotes the following parametric definition, in which X, Y, and Z stand for nets with interface and $n \in \mathbb{N}$.

$$\mathsf{A}_n(\mathsf{X}, \mathsf{Y}, \mathsf{Z}) = hide\Big(\{n{+}1, \ldots, 3n\}, \{1, \ldots, 2n\},\\ add^n_{i=1}\big(n{+}i, 2i{-}1, add^n_{i=1}(2n{+}i, 2i, par(\mathsf{X}, \mathsf{Y}, \mathsf{Z}))\big)\Big).$$

Figure A shows $plus(\vec{p}_0;\vec{t}_0\triangleright N_0,\vec{p}_1;\vec{t}_1\triangleright N_1)$ in the case $|\vec{p}_0| = |\vec{p}_1| = 2$. The named places constitute the resulting input interface, while we show in parentheses the original orderings of $\vec{p}_0$ and $\vec{p}_1$.

As a further example, we specify an idealized scheduler of n tasks, $P_1, \ldots, P_n$, which activates them in cyclic order, with the only constraint that a task cannot be activated again before completing its performance. Following [12, pp. 37], we leave unspecified P_i: we assume that the scheduler fires an α_i-transition (β_i-transition) representing the activation (completion) of P_i. Then, we can proceed as follows. Let $p_1, p_2; t \triangleright C_1$ and $p; t \triangleright C_2$ be the nets shown below, and consider $\mathit{Cell} = \mathit{hide}\big(\{1,3\}, \{2\}, \mathit{add}\big(1,2;3,1, \mathit{par}(C_1, C_2)\big)\big)$.



For $i = 1, \ldots, n$, let Cell_i be $\mathit{rel}\big(\{{}^{\alpha_i}/_{\alpha}, {}^{\beta_i}/_{\beta}\}, \mathit{Cell}\big)$, augment the initial marking of Cell_1 by adding one token to its (unique) interface-place, and define

$$\mathit{Sched} = \mathit{hide}\big(\{1,\ldots,n\}, \{1,\ldots,n\}, \mathit{add}_{i=1}^{n}\big((i \bmod n)+1, i, \mathit{par}_{i=1}^{n}(\mathit{Cell}_i)\big)\big).$$

Writing ϕ and ϕ_i for the relabellings which map to τ all the β_i's, and respectively every label except α_i and β_i, we have that the languages of $\mathit{rel}(\phi, \mathit{Sched})$ and $\mathit{rel}(\phi_i, \mathit{Sched})$ are, respectively, the prefix-closures of $\{\alpha_1 \cdots \alpha_n\}^*$ and of $\{\alpha_i\beta_i\}^*$, which is one way of saying that *Sched* satisfies its specification.

3 $\mathcal{B}$-Behavioural Congruences for Petri Nets

The semantic equivalence of concurrent systems can be described in terms of several kinds of models, e.g., languages, traces, pomsets, event structures, etc., which reflect different assumptions about how system behaviour is to be observed. For nets with interface we consider the following simple cases.

Definition 3.1 (*$\mathcal{B}$-Behavioural Equivalences*)
*Let $\mathcal{B}$: **PN** $\to$ **B** be a function which assigns behaviours to nets in a chosen domain of behaviours **B**. We extend $\mathcal{B}$ to a function **IPN** $\to$ **B** by decreeing that $\mathcal{B}(\vec{p};\vec{t}\triangleright N) = \mathcal{B}(N)$. The $\mathcal{B}$-behavioural equivalence is the equivalence $\approx_{\mathcal{B}}$ induced by $\mathcal{B}$ on **IPN**, viz., $\vec{p}_0;\vec{t}_0 \triangleright N_0 \approx_{\mathcal{B}} \vec{p}_0;\vec{t}_1 \triangleright N_1$ if $\mathcal{B}(\vec{p}_0;\vec{t}_0 \triangleright N_0) = \mathcal{B}(\vec{p}_1;\vec{t}_1 \triangleright N_1)$.*

Following this pattern for the notions of language, step language, and their maximal versions given in Definition 1.3, in correspondence of the functions $\mathcal{L}\colon \mathbf{PN} \to \wp(Act^*)$, $\mathcal{L}_m\colon \mathbf{PN} \to \wp\big(Act^* \cdot \{\epsilon, \checkmark\}\big)$, $\mathcal{S}\colon \mathbf{PN} \to \wp\big(\mu(Act)^*\big)$, and $\mathcal{S}_m\colon \mathbf{PN} \to \wp\big(\mu(Act)^* \cdot \{\epsilon, \checkmark\}\big)$, we find the equivalences $\approx_{\mathcal{L}}$, $\approx_{\mathcal{L}_m}$, $\approx_{\mathcal{S}}$, and $\approx_{\mathcal{S}_m}$, called, respectively, *trace*, *maximal-trace*, *step*, and *maximal-step equivalence.*

Although Definition 3.1 accounts for a greater generality, in the following we shall let $\mathcal{B}$ range only on $\mathcal{L}$, $\mathcal{L}_m$, $\mathcal{S}$, and $\mathcal{S}_m$.

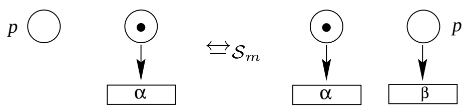
The equivalences $\approx_{\mathcal{B}}$ introduced above are the traditional non-branching equivalences for nets; branching versions can be obtained applying the idea of *bisimulation* [13] to the kinds of behaviour we are considering. We obtain four notions: *(trace) bisimulation* and *maximal-trace bisimulation*, corresponding to $\mathcal{L}$ and $\mathcal{L}_m$, for the interleaving case, and *step* and *maximal-step bisimulation*, corresponding to $\mathcal{S}$ and $\mathcal{S}_m$, for the noninterleaving one.

Definition 3.2 (*$\mathcal{B}$-Bisimulations*)
A (step) bisimulation of N_0 and N_1 is a relation $\mathcal{R} \subseteq \mu(P_{N_0}) \times \mu(P_{N_1})$ such that $s_{N_0}\ \mathcal{R}\ s_{N_1}$, and whenever $s\ \mathcal{R}\ \bar{s}$, then (1) for each firing $s[X\rangle s'$ of a transition (respectively a step) in N_0, there exists a firing sequence (respectively a step sequence) $\bar{s}[Y_1 \cdots Y_n\rangle \bar{s}'$ in N_1 with $s'\ \mathcal{R}\ \bar{s}'$ and $\hat{\lambda}_{N_0}(X) = \hat{\lambda}_{N_1}(Y_1 \cdots Y_n)$, and (2) vice versa exchanging the roles of N_0 and N_1.
A (step) bisimulation $\mathcal{R}$ is a maximal-trace (maximal-step) bisimulation if, in (1) and (2) above, each maximal firing (step) $s[X\rangle s'$ can be matched by a maximal firing (step) sequence $\bar{s}[Y_1 \cdots Y_n\rangle \bar{s}'$.
Nets $\vec{p}_0; \vec{t}_0 \rhd N_0$ and $\vec{p}_1; \vec{t}_1 \rhd N_1$ are (trace), maximal-trace, step, maximal-step bisimilar, written $\vec{p}_0; \vec{t}_0 \rhd N_0 \underline{\leftrightarrow}_{\mathcal{B}}\ \vec{p}_1; \vec{t}_1 \rhd N_1$, $\mathcal{B} \in \{\mathcal{L}, \mathcal{L}_m, \mathcal{S}, \mathcal{S}_m\}$, if there exists a (trace), a maximal-trace, a step, a maximal-step bisimulation of N_0 and N_1.

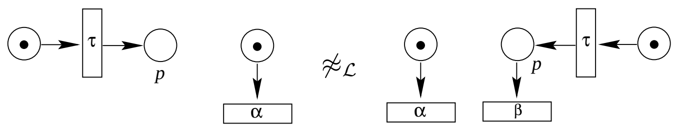
The relationships between the discriminating powers of the equivalences defined above is shown by the following diagram, where $\Rightarrow$, which stands for *strict* set inclusion, follows easily from the definitions.

$$\begin{tikzcd}
 & & \approx_{\mathcal{L}} & & \\
\approx_{\mathcal{L}_m} \arrow[urr, Rightarrow] & & \underline{\leftrightarrow}_{\mathcal{L}} \arrow[u, Rightarrow] & & \approx_{\mathcal{S}} \arrow[ull, Rightarrow] \\
\underline{\leftrightarrow}_{\mathcal{L}_m} \arrow[u, Rightarrow] \arrow[urr, Rightarrow] & & \approx_{\mathcal{S}_m} \arrow[ull, Rightarrow] \arrow[urr, Rightarrow] & & \underline{\leftrightarrow}_{\mathcal{S}} \arrow[u, Rightarrow] \arrow[ull, Rightarrow] \\
 & & \underline{\leftrightarrow}_{\mathcal{S}_m} \arrow[ull, Rightarrow] \arrow[u, Rightarrow] \arrow[urr, Rightarrow] & &
\end{tikzcd}$$

For engineering reasons, related to feasibility of correctness verification for complex systems, for mathematical reasons, related to the availability of equational reasoning, and for more conceptual reasons, related to common intuitions about system equivalence, it is important to consider equivalences which are congruences for a chosen set of system constructors. This guarantees that systems can be replaced by equivalent ones in any context. As shown by the following example, none of the equivalences described above is a congruence for **CM**.



Assuming p in the interface and inserting these nets in the context which connects p with the output of a simple net which can only fire a τ-transitions once, we get different behaviours of the global systems. It should be immediately clear that the problematic combinator is $add(i, j, \_)$.



We are thus led to the largest congruences contained in these equivalences.

Definition 3.3 (*$\mathcal{B}$-Behavioural Congruences*)
For $\approx$ an equivalence on **IPN**, *let $\approx^c$ denote the largest congruence for* **CM** *contained in $\approx$, i.e., $\vec{p}_0;\vec{t}_0 \triangleright N_0 \approx^c \vec{p}_1;\vec{t}_1 \triangleright N_1$ if and only if, for each $\mathcal{C} \in$* **CTX**, *either $\mathcal{C}[\vec{p}_0;\vec{t}_0 \triangleright N_0] = \bot = \mathcal{C}[\vec{p}_1;\vec{t}_1 \triangleright N_1]$ or $\mathcal{C}[\vec{p}_0;\vec{t}_0 \triangleright N_0] \approx \mathcal{C}[\vec{p}_1;\vec{t}_1 \triangleright N_1]$.*

Applying Definition 3.3 to the $\mathcal{B}$-behavioural equivalences we consider, we obtain $\approx^c_{\mathcal{L}}$, $\approx^c_{\mathcal{L}_m}$, $\approx^c_{\mathcal{S}}$, $\approx^c_{\mathcal{S}_m}$, $\underline{\leftrightarrow}^c_{\mathcal{L}}$, $\underline{\leftrightarrow}^c_{\mathcal{L}_m}$, $\underline{\leftrightarrow}^c_{\mathcal{S}}$, and $\underline{\leftrightarrow}^c_{\mathcal{S}}$, which may be called, respectively, *trace*, *maximal-trace*, *step*, *maximal-step*, *bisimulation*, *maximal-trace-bisimulation*, *step-bisimulation*, and *maximal-step-bisimulation congruence*. The relationships between $\approx^c_{\mathcal{B}}$ and $\underline{\leftrightarrow}^c_{\mathcal{B}}$ are analogous to those between the corresponding equivalences, as can be seen by noticing that for subclass of nets with empty interfaces, i.e., for **PN**, $\approx_{\mathcal{B}}$ and $\approx^c_{\mathcal{B}}$, $\underline{\leftrightarrow}_{\mathcal{B}}$ and $\underline{\leftrightarrow}^c_{\mathcal{B}}$ coincide. Thus the global picture can be summarized by the diagram given for $\approx_{\mathcal{B}}$ and $\underline{\leftrightarrow}_{\mathcal{B}}$ and the following diagrams.

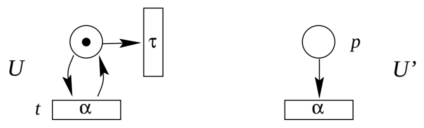
$$\begin{tikzcd}
 & \approx^c_{\mathcal{L}} & \\
\approx^c_{\mathcal{L}_m} \arrow[ur, Rightarrow] & \underline{\leftrightarrow}^c_{\mathcal{L}} \arrow[u, Rightarrow] & \approx^c_{\mathcal{S}} \arrow[ul, Rightarrow] \\
\underline{\leftrightarrow}^c_{\mathcal{L}_m} \arrow[u, Rightarrow] \arrow[ur, Rightarrow] & \approx^c_{\mathcal{S}_m} \arrow[ul, Rightarrow] \arrow[ur, Rightarrow] & \underline{\leftrightarrow}^c_{\mathcal{S}} \arrow[u, Rightarrow] \arrow[ul, Rightarrow] \\
 & \underline{\leftrightarrow}^c_{\mathcal{S}_m} \arrow[ul, Rightarrow] \arrow[u, Rightarrow] \arrow[ur, Rightarrow] &
\end{tikzcd}$$

$$\begin{tikzcd}
 & \approx_{\mathcal{B}} & \\
 & & \underline{\leftrightarrow}_{\mathcal{B}} \arrow[ul, Rightarrow] \\
\approx^c_{\mathcal{B}} \arrow[uur, Rightarrow] & & \\
 & & \underline{\leftrightarrow}^c_{\mathcal{B}} \arrow[ull, Rightarrow] \arrow[uu, Rightarrow]
\end{tikzcd}$$

4 Universal Contexts for $\mathcal{B}$-Behavioural Congruences

In the rest of the paper we focus on characterizing the $\mathcal{B}$-behavioural congruences introduced above. In this section we shall identify a minimal set of contexts which is *universal* for $\approx^c_{\mathcal{B}}$. To make this statement precise, first observe that two nets with interface whose interfaces have different dimensions cannot be congruent, as a context which is undefined only on one of them is easily formulated. Furthermore, for each $\vec{p};\vec{t} \triangleright N$, we provide in the following *one* context—determined by the dimensions of $\vec{p};\vec{t}$—which separates $\vec{p};\vec{t} \triangleright N$ from all and only the nets which are '$\mathcal{B}$-separable' from it by some context in **CTX**, i.e., the nets not $\approx_{\mathcal{B}}$-congruent to it, and which have interfaces of the same dimensions as $\vec{p};\vec{t}$. All this means that to determine whether two nets are $\approx_{\mathcal{B}}$-congruent, it is enough to verify that the numbers of places and transitions in the respective interfaces coincide and, in case, to compare their $\mathcal{B}$-behaviours into a single 'universal' context. This result can be extended to $\leftrightarrows^{c}_{\mathcal{B}}$, provided we restrict ourselves to *well-labelled* nets with interface, where $p_1,\ldots,p_n;t_1,\ldots,t_m \rhd N$ is well-labelled if $\lambda_N(t_i) \neq \tau$, for $i = 1,\ldots,m$. Observe that, in view of the intuition of interface-transitions as public 'output' channels, this is a rather mild and reasonable condition to impose on nets with interface.

Recalling that *Act* is equipped with an enumeration $\alpha_1,\alpha_2,\ldots$, let ψ be the relabelling function $\alpha_i \mapsto \alpha_{3i}$, $i \in \mathbb{N}$. Let $\varnothing;t \rhd U$ and $p;\varnothing \rhd U'$ be the nets with interface shown in the figure below.



Definition 4.1 (*Universal Contexts*)
Let $\mathcal{C}_{i,j}$ and $\mathcal{U}_{i,j}$, $i,j \in \mathbb{N}$, be the contexts defined below.

$$\begin{aligned}\mathcal{C}_{i,j} &= par\Big(par^{i}_{k=1}\big(\varnothing;t \rhd U[^{\alpha_{3k-2}}/_{\alpha}]\big), par^{j}_{k=1}\big(p;\varnothing \rhd U'[^{\alpha_{3k-1}}/_{\alpha}]\big)\Big),\\ \mathcal{U}_{i,j} &= add^{j}_{k=1}\Big(k, i{+}k, add^{i}_{k=1}\Big(j{+}k, k, par\big(\mathcal{C}_{i,j}, rel(\psi,\_)\big)\Big)\Big).\end{aligned}$$

Figure B presents $\mathcal{U}_{i,j}[\vec{p};\vec{t} \rhd N]$ for a $\vec{p};\vec{t} \rhd N$ with $|\vec{p}| = i$ and $|\vec{t}| = j$. The interface of $\mathcal{U}_{i,j}[\vec{p};\vec{t} \rhd N]$ is shown by naming and numbering the places and transitions which belong to it. The information in parentheses concern the orderings in $\vec{p};\vec{t}$. Concerning the labels, we use ι_k for α_{3k-2}, $k = 1,\ldots,i$, o_k for α_{3k-1}, $k = 1,\ldots,j$, and $\alpha_{k_1},\ldots\alpha_{k_j}$ for the labels of $\vec{t}$ in N. The dashed arrows are those inserted by *add*.

The contexts $\mathcal{U}_{i,j}$ are conceptually very simple. They provide a copy of $\varnothing;t \rhd U$ for each place in $\vec{p}$, and a copy of $p;\varnothing \rhd U'$ for each transition in $\vec{t}$. The cascade of $add(i,j,\_)$ combinators connects together the transition-place pairs so created. The role of the collection of $\varnothing;t \rhd U$ is to test the 'reactivity' of the 'input' sites of $\vec{p};\vec{t} \rhd N$ by sending in any number of tokens, at any relative speed, and independently for any place in $\vec{p}$. The collection of $p;\varnothing \rhd U'$ tests the 'output'-behaviour by recording the occurrences of the transitions in $\vec{t}$.

In order for these contexts to form universal collections, we need to be able to distinguish in the behaviour of $\mathcal{U}_{i,j}[\vec{p};\vec{t} \rhd N]$ the actions stemming from $\mathcal{U}_{i,j}$ from those stemming from N. This is achieved using the $rel(\psi,\_)$ combinator: since the actions of N are uniformly 'remapped' to $3k$-numbered actions, we are free to use differently numbered actions in the contexts. The soundness of this technique relies on the fact that ψ is injective and, therefore, no equivalences are enforced by the ψ-relabelling.

We can now make precise our statements about the universality of the contexts $\mathcal{U}_{i,j}$. We start by defining eight equivalences on **IPN**, viz., $\approx^{u}_{\mathcal{B}}$ and $\leftrightarrows^{u}_{\mathcal{B}}$

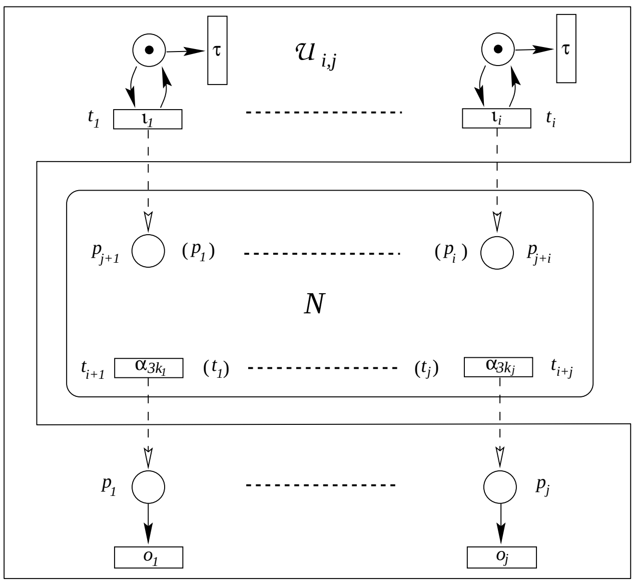


Figure B

for $\mathcal{B} \in \{\mathcal{L}, \mathcal{L}_m, \mathcal{S}, \mathcal{S}_m\}$, which take into account *only* the behaviour inside one universal context, chosen depending on the size of the interface of the involved nets. Theorem 4.3 will establish that such '$\approx_{\mathcal{B}}$-universal' equivalences coincide, respectively, with trace, maximal-trace, step, and maximal step congruences. Finally, Theorem 4.4 extends the result to the '$\leftrightarrows_{\mathcal{B}}$-universal' equivalences and bisimulation and step-bisimulation congruences in the case of well-labelled nets with interface.

Due to the extended abstract nature of this exposition and the space limitation, here and in the following section all the proofs will be omitted.

Definition 4.2 (*$\mathcal{B}$-Universal Equivalences*)
For $\mathcal{B} \in \{\mathcal{L}, \mathcal{L}_m, \mathcal{S}, \mathcal{S}_m\}$, let $\approx^u_{\mathcal{B}}$ be the equivalence $\lesssim^u_{\mathcal{B}} \cap \gtrsim^u_{\mathcal{B}}$, where $\lesssim^u_{\mathcal{B}}$ denotes the following preorder on **IPN**.

$$\vec{p}_0;\vec{t}_0 \rhd N_0 \lesssim^u_{\mathcal{B}} \vec{p}_1;\vec{t}_1 \rhd N_1 \;\Leftrightarrow\; |\vec{p}_0| = |\vec{p}_1| = i,\ |\vec{t}_0| = |\vec{t}_1| = j,\ \text{and}\ \mathcal{B}(\mathcal{U}_{i,j}[\vec{p}_0;\vec{t}_0 \rhd N_0]) \subseteq \mathcal{B}(\mathcal{U}_{i,j}[\vec{p}_1;\vec{t}_1 \rhd N_1]).$$

For $\mathcal{B} \in \{\mathcal{L}, \mathcal{L}_m, \mathcal{S}, \mathcal{S}_m\}$, let $\leftrightarrows^u_{\mathcal{B}}$ be the following equivalence on **IPN**.

$$\vec{p}_0;\vec{t}_0 \rhd N_0 \leftrightarrows^u_{\mathcal{B}} \vec{p}_1;\vec{t}_1 \rhd N_1 \;\Leftrightarrow\; |\vec{p}_0| = |\vec{p}_1| = i,\ |\vec{t}_0| = |\vec{t}_1| = j,\ \text{and}\ \mathcal{U}_{i,j}[\vec{p}_0;\vec{t}_0 \rhd N_0] \leftrightarrows_{\mathcal{B}} \mathcal{U}_{i,j}[\vec{p}_1;\vec{t}_1 \rhd N_1].$$

Theorem 4.3 (*The $\mathcal{U}_{i,j}$'s are Universal, I*)
$\approx^u_{\mathcal{L}} = \approx^c_{\mathcal{L}}$, $\approx^u_{\mathcal{L}_m} = \approx^c_{\mathcal{L}_m}$, $\approx^u_{\mathcal{S}} = \approx^c_{\mathcal{S}}$, *and* $\approx^u_{\mathcal{S}_m} = \approx^c_{\mathcal{S}_m}$.

Theorem 4.4 (*The $\mathcal{U}_{i,j}$'s are Universal, II*)
For $\mathcal{B} \in \{\mathcal{L}, \mathcal{L}_m, \mathcal{S}, \mathcal{S}_m\}$, and $\vec{p}_0;\vec{t}_0 \rhd N_0$ and $\vec{p}_1;\vec{t}_1 \rhd N_1$ well-labelled nets with interface, $\vec{p}_0;\vec{t}_0 \rhd N_0 \leftrightarrows^c_{\mathcal{B}} \vec{p}_1;\vec{t}_1 \rhd N_1$ if and only if $\vec{p}_0;\vec{t}_0 \rhd N_0 \leftrightarrows^u_{\mathcal{B}} \vec{p}_1;\vec{t}_1 \rhd N_1$.

5 A Game Theoretic Approach to $\mathcal{B}$-Behavioural Congruences

In this section we sketch game theoretic characterizations of the $\mathcal{B}$-behavioural congruences introduced in the paper. The results here are simply 're-readings' in terms of games of Theorems 4.3 and 4.4. The exposition will be rather informal; the reader is referred to [1, 22] for a formal presentation of games.

Notation. For simplicity, in this section we shall assume that $Act \cap (Act \times \mathbb{N}) = \varnothing$. For $\vec{p}; \vec{t} \triangleright N$ a net with interface, let $\lambda_N^{\vec{t}}$ be the modification of λ_N which records explicitly the occurrences of interface-transitions by assigning the pair $\langle \lambda_N(t), k\rangle$ to $t = t_k \in \vec{t}$ and simply $\lambda_N(t)$ to $t \notin \vec{t}$. As in Definition 1.3, we use $\hat{\lambda}_N^{\vec{t}}$ for the obvious extension of $\lambda_N^{\vec{t}}$ to steps. Notice that $\hat{\lambda}_N^{\vec{t}}$ disregards the τ-transitions, but not the $\langle \tau, k\rangle$-transitions.

We start by describing the game $\mathsf{G}_{\mathcal{L}}$. The game is played on two nets with interface $\vec{p}_0; \vec{t}_0 \triangleright N_0$ and $\vec{p}_1; \vec{t}_1 \triangleright N_1$ such that $|\vec{p}_0| = |\vec{p}_1| = i$ and $|\vec{t}_0| = |\vec{t}_1| = j$. There are two players: Player, who plays with $\vec{p}_0; \vec{t}_0 \triangleright N_0$, and Opponent, who plays with $\vec{p}_1; \vec{t}_1 \triangleright N_1$. At any round $k \geq 1$ of the game, Player 0 starts from a state s_k of N_0, whilst Opponent maintains a finite set S_k of states of N_1; assume $s_0 = s_{N_0}$ and $S_0 = \{s_{N_1}\}$. Round k is played as follows.

- ▷ Player selects i natural numbers $\vec{n}_k$, updates s_{k-1} to s'_k by inserting $\vec{n}_k$ tokens in the places $\vec{p}_0$, fires a *transition* $s'_k[t\rangle s_k$ of N_0, and presents $\hat{\lambda}_{N_0}^{\vec{t}_0}(t)$ to Opponent. (I.e., Player shows only the visible actions, but *declares* the occurrence of transitions in $\vec{t}_0$.)
- ▷ Opponent answers by selecting $\bar{s} \in S_{k-1}$, updating it to $\bar{s}'$ with the insertion of $\vec{n}_k$ tokens in the places $\vec{p}_1$, and by producing a *firing sequence* $\bar{s}'[Y_1 \cdots Y_m\rangle \bar{s}''$ of N_1 such that $\hat{\lambda}_{N_1}^{\vec{t}_1}(Y_1 \cdots Y_m) = \hat{\lambda}_{N_0}^{\vec{t}_0}(t)$. Then, Opponent chooses S_k as a finite set of states reachable from a state in S_{k-1} in the way $\bar{s}''$ is reached from $\bar{s}$.
- ▷ If Opponent cannot answer, then Player *wins* the game. Otherwise, the players start round $k+1$.

Opponent *wins* if after round k Player reaches a state s_k in which no transitions are fireable, or if the game does not terminate. In the following we shall also consider three simple variations of $\mathsf{G}_{\mathcal{L}}$.

Var1: Instead of *transitions* and *firing sequences*, the players play, respectively, with *steps* and *step sequences*.

Var2: If at round k Player reaches a state s_k in which no transition of N_0 can fire, then Opponent wins the game *only* if there exists $\bar{s}_k \in S_k$ in which no transition of N_1 is fireable. Otherwise, Player wins.

Var3: Opponent plays with only *one* state, and at each round Player chooses whether to play the round on $\vec{p}_0; \vec{t}_0 \triangleright N_0$ or on $\vec{p}_1; \vec{t}_1 \triangleright N_1$.

We write $\mathsf{G}_{\mathcal{L}_m}$ for $\mathsf{G}_{\mathcal{L}}$ with Var2, $\mathsf{G}_{\mathcal{S}}$ for $\mathsf{G}_{\mathcal{L}}$ with Var1, and $\mathsf{G}_{\mathcal{S}_m}$ for $\mathsf{G}_{\mathcal{L}}$ with Var1 and Var2. Moreover, $\mathsf{GB}_{\mathcal{B}}$ stands for $\mathsf{G}_{\mathcal{B}}$ with Var3. For $\mathsf{G}_{\mathcal{B}}$ and $\mathsf{GB}_{\mathcal{B}}$ we have the following simple results.

Theorem 5.1 (*Game Characterization of* $\approx_{\mathcal{B}}^{c}$)
For $\mathcal{B} \in \{\mathcal{L}, \mathcal{L}_m, \mathcal{S}, \mathcal{S}_m\}$, Opponent has a winning strategy for $\mathsf{G}_{\mathcal{B}}$ played on $\vec{p}_0; \vec{t}_0 \triangleright N_0$ and $\vec{p}_1; \vec{t}_1 \triangleright N_1$ if and only if $\vec{p}_0; \vec{t}_0 \triangleright N_0 \lesssim_{\mathcal{B}}^{u} \vec{p}_1; \vec{t}_1 \triangleright N_1$.

In other words, $\vec{p}_0;\vec{t}_0 \triangleright N_0 \approx^c_{\mathcal{B}} \vec{p}_1;\vec{t}_1 \triangleright N_1$ if and only if Opponent has a winning strategy for $\mathsf{G}_{\mathcal{B}}$, whether he plays with $\vec{p}_0;\vec{t}_0 \triangleright N_0$ or with $\vec{p}_1;\vec{t}_1 \triangleright N_1$

Theorem 5.2 (*Game Characterization of* $\leftrightarrows^c_{\mathcal{B}}$)
For $\mathcal{B} \in \{\mathcal{L}, \mathcal{L}_m, \mathcal{S}, \mathcal{S}_m\}$, *Opponent has a winning strategy for* $\mathsf{GB}_{\mathcal{B}}$ *played on* $\vec{p}_0;\vec{t}_0 \triangleright N_0$ *and* $\vec{p}_1;\vec{t}_1 \triangleright N_1$ *if and only if* $\vec{p}_0;\vec{t}_0 \triangleright N_0 \leftrightarrows^u_{\mathcal{B}} \vec{p}_1;\vec{t}_1 \triangleright N_1$.

Therefore, for $\vec{p}_0;\vec{t}_0 \triangleright N_0$ and $\vec{p}_1;\vec{t}_1 \triangleright N_1$ *well-labelled* nets with interface, $\vec{p}_0;\vec{t}_0 \triangleright N_0 \leftrightarrows^c_{\mathcal{B}} \vec{p}_1;\vec{t}_1 \triangleright N_1$ if and only if Opponent has a winning strategy for $\mathsf{GB}_{\mathcal{B}}$ played on $\vec{p}_0;\vec{t}_0 \triangleright N_0$ and $\vec{p}_1;\vec{t}_1 \triangleright N_1$.

Conclusion and Future Work

We introduced a simple notion of interface for Petri nets and, based on this, a minimal set **CM** of combinators of nets with interface, which looks promising expressive while remaining very manageable. The key to the tractability of **CM** is the input/output partition of interfaces, which formalizes the intuition that interaction consists of communicating along channels. Moreover, we studied eight behavioural congruences derived from a choice of four simple behavioural notions, and we characterized them via universal contexts and games.

We believe that this work opens some interesting questions, which we must leave for future work. Among them we single out the following ones.

Bisimulations. Work finalized to characterize the **CM**-congruences based on weak bisimulation in the case of non-well-labelled nets is ongoing.

Expressiveness. Adding selected sets of 'constants' to our combinators gives rise to quite powerful *'syntactic' calculi of nets.* E.g., exploiting the results from MIT mentioned in the introduction, we know that such languages can be rich enough to express, *up to* $\approx^c_{\mathcal{B}}$, all finite Petri nets. The interesting question we plan to study is whether this can be achieved also for bisimulations.

Denotational Models. Theorems 4.3, 4.4, 5.1, and 5.2 seem to provide ground for defining semantic models for nets with interface *fully abstract* with respect to $\approx^c_{\mathcal{B}}$ and $\leftrightarrows^c_{\mathcal{B}}$. For $\approx^c_{\mathcal{B}}$ the issue is addressed in the full version of this paper [17].

Action Structures. In retrospect, the algebra presented here has strong analogies, but also non trivial differences, with Milner's (*reflexive*) *action calculus of nets* [14]. We plan to study such connections in further details.

Acknowledgements. We thank Eike Best, Egon Börger, Andrea Corradini, Pierpaolo Degano, and Ugo Montanari for valuable discussions on the argument of this paper. Many thanks to Edmund Robinson for his insightful comments on an earlier version of this paper.

References

[1] S. Abramsky, and R. Jagadeesan, *"Games and Full Completeness for Multiplicative Linear Logic"*, Technical Report DoC 92/24, Imperial College, 1992.

[2] E. Best, R. Devillers, and J. Hall, "The Petri Box Calculus: a New Causal Algebra with Multilabel Communication", in *Advances in Petri Nets*, G. Rozenberg, Ed., LNCS n. 609, pp. 21–69, Springer-Verlag, 1992.

[3] C. Brown, D. Gurr, and V. de Paiva, *"A Linear Specification Language for Petri Nets"*, Report DAIMI PB-363, Computer Science Dept., University of Aarhus, 1991.

[4] J. Bruno, and S.M. Altman, "A Theory of Asynchronous Control Networks", *IEEE Transactions on Computers*, Vol. C-20, pp. 629–638, 1971.

[5] P. Degano, J. Meseguer, and U. Montanari, "Axiomatizing Net Computations and Processes", in *Proceedings of the 4th LICS Symposium*, pp. 175–185, IEEE Press, 1989.

[6] J.B. Dennis, "Modular, Asynchronous Control Structures for a High Performance Processor", in *Record of the Project MAC Conference on Concurrent Systems and Parallel Computation*, ACM Press, 1970.

[7] R. Gorrieri, and U. Montanari, "Scone: A Simple Calculus of Nets", in *Proceedings of CONCUR* '90, LNCS n. 458, J. Baeten, J. Klop, Eds., pp. 2–31, Springer-Verlag, 1990.

[8] M. Hennessy, *"Algebraic Theory of Processes"*. MIT Press, 1988.

[9] C.A.R. Hoare, *"Communicating Sequential Processes"*. Prentice Hall, 1985.

[10] R.M. Keller, "Towards a Theory of Universal Speed-Independent Modules", *IEEE Transactions on Computers*, Vol. C-23, pp. 21–33, 1974.

[11] J. Meseguer, and U. Montanari, "Petri Nets are Monoids", *Information and Computation*, n. 88, Academic Press, pp. 105–154, 1990.

[12] R. Milner, *"A Calculus of Communicating Systems"*. LNCS n. 92, Springer-Verlag, 1980.

[13] R. Milner, *"Communication and Concurrency"*. Prentice Hall, 1989.

[14] R. Milner, "Action Calculi or Syntactic Action Structures", in *Proceedings of MFCS* '93, LNCS n. 711, A. Borzyszkowski, S. Sokolowski, Eds., pp. 105–121, Springer-Verlag, 1993.

[15] M. Nielsen, and C. Clausen, "Bisimulation for Models in Concurrency", in *Proceedings of CONCUR '94*, LNCS n. 836, B. Jonsson, J. Parrow, Eds., pp. 385–400, Springer-Verlag, 1994.

[16] M. Nielsen, G. Plotkin, and G. Winskel, "Petri Nets, Event Structures and Domains, Part 1", *Theoretical Computer Science*, n. 13, Elsevier, pp. 85–108, 1981.

[17] M. Nielsen, L. Priese, and V. Sassone, *"An Algebra of Petri Nets"*, To appear as Technical Report BRICS, RS Series, 1995.

[18] S.S. Patil, *"Limitations and Capabilities of Dijkstra's Semaphore Primitives for Coordination among Processes"*, Computation Structure Group Memo n. 57, Project MAC, MIT, 1971.

[19] L. Priese, "Automata and Concurrency", *Theoretical Computer Science*, n. 25, Elsevier, pp. 221–265, 1983.

[20] W. Reisig, *"Petri Nets"*. Springer-Verlag, 1985.

[21] V. Sassone, "On the Category of Petri Net Computations", in *Proceedings of TAPSOFT* '95, P. Mosses, M. Nielsen, M.I. Schwartzbach, Eds., LNCS n. 915, pp. 334–348, Springer-Verlag, 1995.

[22] C. Stirling, *"Modal and Temporal Logics for Processes"*, Notes for Summer School in Logic Methods in Concurrency, Computer Science Dept., University of Aarhus, 1993.

[23] G. Winskel, "Petri Nets, Algebras, Morphisms and Compositionality", *Information and Computation*, n. 72, Academic Press, pp. 197–238, 1987,

[24] M. Yoeli, *"Petri Nets and Asynchronous Control Networks"*, Report CS–73–07, Dept. of Applied Mathematics and Computer Science, University of Waterloo, 1973.