Fundamenta Informaticae 190(2-4) : 159–207 (2024)
Available at IOS Press through:
https://doi.org/10.3233/FI-242169

Correctness Notions for Petri Nets with Identifiers

Jan Martijn E.M. van der Werf *
Utrecht University
Princetonplein 5, 3584 CC Utrecht
The Netherlands
j.m.e.m.vanderwerf@uu.nl

Andrey Rivkin
Technical University of Denmark
Richard Petersens Plads 321
2800 Kgs., Lyngby, Denmark
ariv@dtu.dk

Marco Montali
Free University of Bozen-Bolzano
piazza Domenicani 3
39100 Bolzano, Italy
montali@inf.unibz.it

Artem Polyvyanyy
The University of Melbourne
Grattan Street, Parkville
Victoria, 3010, Australia
artem.polyvyanyy@unimelb.edu.au

Abstract. A model of an information system describes its processes and how resources are involved in these processes to manipulate data objects. This paper presents an extension to the Petri nets formalism suitable for describing information systems in which states refer to object instances of predefined types and resources are identified as instances of special object types. Several correctness criteria for resource- and object-aware information systems models are proposed, supplemented with discussions on their decidability for interesting classes of systems. These new correctness criteria can be seen as generalizations of the classical soundness property of workflow models concerned with process control flow correctness.

Keywords: Information System, Verification, Data Correctness, Resource Correctness

1. Introduction

Petri nets are widely used to describe distributed systems capable of expanding their resources indefinitely [1]. A *Petri net* describes passive and active components of a system, modeled as places and transitions, respectively. The active components of a Petri net communicate asynchronously with each

*Address for correspondence: Utrecht University, Princetonplein 5, 3584 CC Utrecht, The Netherlands.

other via local interfaces. Thus, state changes in a Petri net system have local causes and effects and are modeled as tokens consumed, produced, or transferred by the transitions of the system. A *token* is often used to denote an *object* in the physical world the system manipulates or a *condition* that can cause a state change in the system.

Petri nets with identifiers [2] extend classical Petri nets to provide formal means to relate tokens to objects. Every token in such a Petri net is associated with a vector of identifiers, where each identifier uniquely identifies a data object. Consequently, active components of a Petri net with identifiers model how groups of objects, either envisioned or those existing in the physical world, can be consumed, produced, or transferred by the system.

It is often desirable that modeled systems are correct. Many criteria have been devised for assessing the correctness of systems captured as Petri nets. Those criteria target models of systems that use tokens to represent conditions that control their state changes. In other words, they can be used to verify the correctness of processes the systems can support and not of the object manipulations carried out within those processes. Such widely-used criteria include boundedness [3], liveness [4], and soundness [5]. The latter one, for instance, ensures that a system modeled as a *workflow net*, a special type of Petri nets used to encode workflows at organizations, has a terminal state that can be distinguished from other states of the system, the system can always reach the terminal state, and every transition of the system can in principle be enabled and, thus, used by the system.

Real-world systems, such as information systems [6], are characterized by processes that manipulate objects. For instance, an online retailer system manipulates products, invoices, and customer records. However, although tools allow designing such models [7], initial use showed that correctness criteria addressing both aspects, that is, the processes and data, are understood less well [8]. The paper at hand closes this gap.

In this paper, we propose a correctness criterion for Petri nets with identifiers that combines the checks of the soundness of the system's processes with the soundness of object manipulations within those processes. Intuitively, objects of a specific type are correctly manipulated by the system if every object instance of that type, characterized by a unique identifier, can "leave" the system, that is, a dedicated transition of the system can consume it, and once that happens, no references to that object instance remain in the system. When a system achieves this harmony for its processes and all data object types, we say that the system is *identifier sound*, or, alternatively, that the data and processes of the system are in *resonance*. Specifically, this paper makes these contributions:

- It motivates and defines the notion of *identifier soundness* for checking correctness of data object manipulations in processes of a system;
- It proposes a resource-aware extension for systems and defines a suitable correctness criterion building on top of the one of identifier soundness and requiring that system resources are managed *conservatively*;
- It discusses aspects related to *decidability of identifier soundness* in the general case and for certain restricted, but still useful, classes of systems;
- It establishes connections with existing results on verification of data-aware processes and shows which verification tasks are decidable for object-aware systems.

The paper proceeds as follows. The next section introduces concepts and notions required to support subsequent discussions. Section 3 introduces typed Petri nets with identifiers, a model for modeling distributed systems whose state is defined by objects the system manipulates. Section 4 presents various correctness notions for typed Petri nets with identifiers, including identifier soundness, and demonstrates a proof that the notion is in general undecidable. Moreover, the section discusses the connection to existing verification results and shows which verification tasks are decidable for typed Petri nets with identifiers. Section 5 discusses several classes of systems for which identifier soundness is guaranteed by construction. Section 6 presents the formalism extension with resource management capabilities and discusses a series of results, including resource-aware soundness, that is deemed to be undecidable. Finally, the paper concludes with a discussion of related work and future work.

2. Preliminaries

Let S and T be sets. The powerset of S is denoted by $\wp(S) = \{S' \mid S' \subseteq S\}$ and $|S|$ denotes the cardinality of S. Given a relation $R \subseteq S \times T$, its range is defined by $\text{RNG}(R) = \{y \in T \mid \exists x \in S : (x, y) \in R\}$. [1] A *multiset* m over S is a mapping of the form $m : S \to \mathbb{N}$, where $\mathbb{N} = \{0, 1, 2, \ldots\}$ denotes the set of natural numbers. For $s \in S$, $m(s) \in \mathbb{N}$ denotes the number of times s appears in the multiset. For $x \notin S$, $m(x) = 0$. We write s^n if $m(s) = n$. We use $S^{\oplus}$ to denote the set of all finite multisets over S and $\emptyset$ to denote the *empty multiset*. The support of $m \in S^{\oplus}$ is the set of elements that appear in m at least once: $supp(m) = \{s \in S \mid m(s) > 0\}$. Given two multisets m_1 and m_2 over S, we consider the following standard multiset operations:

- $m_1 \leq m_2$ iff $m_1(s) \leq m_2(s)$ for each $s \in S$;
- $m_1 + m_2 = \{s^n \mid s \in S, n = m_1(s) + m_2(s)\}$;
- if $m_1 \leq m_2$, $m_2 - m_1 = \{s^n \mid s \in S, n = m_2(s) - m_1(s)\}$.

We also write $|m| = \sum_{s \in S} m(s)$ to denote the cardinality of m. A *sequence* over S of length $n \in \mathbb{N}$ is a function $\sigma : \{1, \ldots, n\} \to S$. If $n > 0$ and $\sigma(i) = a_i$, for $1 \leq i \leq n$, we write $\sigma = \langle a_1, \ldots, a_n \rangle$. The length of σ is denoted by $|\sigma|$ and is equal to n. The sequence of length 0 is called the *empty sequence*, and is denoted by ϵ. The set of all finite sequences over S is denoted by S^*. We write $a \in \sigma$ if there is $1 \leq i \leq |\sigma|$ such that $\sigma(i) = a$. *Concatenation* of two sequences $\nu, \gamma \in S^*$, denoted by $\sigma = \nu \cdot \gamma$, is a sequence defined by $\sigma : \{1, \ldots, |\nu| + |\gamma|\} \to S$, such that $\sigma(i) = \nu(i)$ for $1 \leq i \leq |\nu|$, and $\sigma(i) = \gamma(i - |\nu|)$ for $|\nu| + 1 \leq i \leq |\nu| + |\gamma|$. We define the projection of sequences on a set T by induction as follows: *(i)* $\epsilon_{|T} = \epsilon$; *(ii)* $(\langle a \rangle \cdot \sigma)_{|T} = \langle a \rangle \cdot \sigma_{|T}$, if $a \in T$; *(iii)* $(\langle a \rangle \cdot \sigma)_{|T} = \sigma_{|T}$, if $a \notin T$. Renaming sequence σ with an injective function $r : S \to T$ is defined inductively by $\rho_r(\epsilon) = \epsilon$, and $\rho_r(\langle a \rangle \cdot \sigma) = \langle r(a) \rangle \cdot \rho_r(\sigma)$. Renaming is extended to multisets of sequences as follows: given a multiset $m \in (S^*)^{\oplus}$, we define $\rho_r(m) = \sum_{\sigma \in supp(m)} \sigma(m) \cdot \rho_r(\sigma)$. For example, $\rho_{\{x \mapsto a, y \mapsto b\}}(\langle x, y \rangle^3) = \langle a, b \rangle^3$.

Labeled Transition Systems. To model the behavior of a system, we use *labeled transition systems*. Given a finite set A of (action) labels, a *(labeled) transition system* (LTS) over A is a tuple $\Gamma = (S, A, s_0, \rightarrow)$, where S is the (possibly infinite) set of *states*, s_0 is the *initial state* and

[1] Notice that R can be also seen as a function $R : X \to T$.

$\rightarrow \subseteq (S \times (A \cup \{\tau\}) \times S)$ is the *transition relation*, where $\tau \notin A$ denotes the silent action [9]. In what follows, we write $s \xrightarrow{a} s'$ for $(s, a, s') \in \rightarrow$. Let $r : A \rightarrow (A' \cup \{\tau\})$ be a total function. Renaming Γ with r is defined as $\rho_r(\Gamma) = (S, A' \cup \{\tau\}, s_0, \rightarrow')$ with $(s, r(a), s') \in \rightarrow'$ iff $(s, a, s') \in \rightarrow$. Given a set T, hiding is defined as $\hat{\mathrm{H}}_T(\Gamma) = \rho_h(\Gamma)$ with $h : A \rightarrow A \cup \{\tau\}$ such that $h(t) = \tau$ if $t \in T$ and $h(t) = t$ otherwise. Given $a \in A$, $p \xrightarrow{a}\!\!\!\!\rightarrow q$ denotes a *weak transition relation* that is defined as follows: *(i)* $p \xrightarrow{a}\!\!\!\!\rightarrow q$ iff $p(\xrightarrow{\tau})^* q_1 \xrightarrow{a} q_2(\xrightarrow{\tau})^* q$; *(ii)* $p \xrightarrow{\tau}\!\!\!\!\rightarrow q$ iff $p(\xrightarrow{\tau})^* q$. Here, $(\xrightarrow{\tau})^*$ denotes the reflexive and transitive closure of $\xrightarrow{\tau}$.

Definition 2.1. (Strong and weak bisimulation)
Let $\Gamma_1 = (S_1, A, s_{01}, \rightarrow_1)$ and $\Gamma_2 = (S_2, A, s_{02}, \rightarrow_2)$ be two LTSs. A relation $R \subseteq (S_1 \times S_2)$ is called a *strong simulation*, denoted as $\Gamma_1 \prec_R \Gamma_2$, if for every pair $(p, q) \in R$ and $a \in A \cup \{\tau\}$, it holds that if $p \xrightarrow{a}_1 p'$, then there exists $q' \in S_2$ such that $q \xrightarrow{a}_2 q'$ and $(p', q') \in R$. Relation R is a *weak simulation*, denoted by $\Gamma_1 \preccurlyeq_R \Gamma_2$, iff for every pair $(p, q) \in R$ and $a \in A \cup \{\tau\}$ it holds that if $p \xrightarrow{a}_1 p'$, then either $a = \tau$ and $(p', q) \in R$, or there exists $q' \in S_2$ such that $q \xrightarrow{a}\!\!\!\!\rightarrow_2 q'$ and $(p', q') \in R$.

R is called a strong (weak) *bisimulation*, denoted by $\Gamma_1 \sim_R \Gamma_2$ ($\Gamma_1 \approx_R \Gamma_2$) if both $\Gamma_1 \prec \Gamma_2$ ($\Gamma_1 \preccurlyeq_R \Gamma_2$) and $\Gamma_2 \prec_{R^{-1}} \Gamma_1$ ($\Gamma_2 \preccurlyeq_{R^{-1}} \Gamma_1$). The relation is called *rooted* iff $(s_{01}, s_{02}) \in R$. A rooted relation is indicated with a superscript ^{r}.

Petri nets. A weighted Petri net is a 4-tuple (P, T, F, W) where P and T are two disjoint sets of *places* and *transitions*, respectively, $F \subseteq ((P \times T) \cup (T \times P))$ is the *flow relation*, and $W : ((P \times T) \cup (T \times P)) \rightarrow \mathbb{N}$ is a *weight function* such that $W(f) > 0$ iff $f \in F$. For $x \in P \cup T$, we write ${}^\bullet x = \{y \mid (y, x) \in F\}$ to denote the *preset* of x and $x^\bullet = \{y \mid (x, y) \in F\}$ to denote the *postset* of x. We lift the notation of preset and postset to sets element-wise. If for a Petri net no weight function is explicitly defined, we assume $W(f) = 1$ for all $f \in F$. A *marking* of N is a multiset $m \in P^\oplus$, where $m(p)$ denotes the number of *tokens* in place $p \in P$. If $m(p) > 0$, place p is called *marked* in marking m. A *marked Petri net* is a tuple (N, m) with N a weighted Petri net with marking m. A transition $t \in T$ is enabled in (N, m), denoted by $(N, m)[t\rangle$ iff $W((p, t)) \leq m(p)$ for all $p \in {}^\bullet t$. An enabled transition can *fire*, resulting in marking m' iff $m'(p) + W((p, t)) = m(p) + W((t, p))$, for all $p \in P$, and is denoted by $(N, m)[t\rangle(N, m')$. We lift the notation of firings to sequences. A sequence $\sigma \in T^*$ is a *firing sequence of* (N, m_0) iff $\sigma = \epsilon$, or markings $m_0, \ldots, m_n$ exist such that $(N, m_{i-1})[\sigma(i)\rangle(N, m_i)$ for $1 \leq i \leq |\sigma| = n$, and is denoted by $(N, m_0)[\sigma\rangle(N, m_n)$. If the context is clear, we omit N, and just write $m_0[\sigma\rangle m_n$. The set of reachable markings of (N, m) is defined by $\mathcal{R}(N, m) = \{m' \mid \exists \sigma \in T^* : m[\sigma\rangle m'\}$. The semantics of a marked Petri net (N, m_0) with $N = (P, T, F, W)$ is defined by the LTS $\Gamma_{N,m_0} = (P^\oplus, T, m_0, \rightarrow)$ with $(m, t, m') \in \rightarrow$ iff $m[t\rangle m'$.

Workflow Nets. A *workflow net* (WF-net for short) is a tuple $N = (P, T, F, W, in, out)$ such that: *(i)* (P, T, F, W) is a weighted Petri net; *(ii)* $in, out \in P$ are the source and sink place, respectively, with ${}^\bullet in = out^\bullet = \emptyset$; *(iii)* every node in $P \cup T$ is on a directed path from in to out. N is called k-*sound* for some $k \in \mathbb{N}$ iff *(i)* it is proper completing, i.e., for all reachable markings $m \in \mathcal{R}(N, [in^k])$, if $[out^k] \leq m$, then $m = [out^k]$; *(ii)* it is weakly terminating, i.e., for any reachable marking $m \in \mathcal{R}(N, [in^k])$, the final marking is reachable, i.e., $[out^k] \in \mathcal{R}(N, m)$; and *(iii)* it is quasi-live, i.e., for all transitions $t \in T$, there is a marking $m \in \mathcal{R}(N, [in])$ such that $m[t\rangle$. The net is called *sound* if it is 1-sound. If it is k-sound for all $k \in \mathbb{N}$, it is called *generalized sound* [10].

3. Typed Petri nets with identifiers

Processes and data are highly intertwined: processes manipulate data objects while objects govern processes. For example, consider a retail shop with three types of objects: *products* sold through the shop, *customers* that can order these products, and *orders* that track products bought by customers. This example already involves many-to-many relations between objects, e.g., a product can be ordered by many customers, while a customer can order many products. Relations between objects can also be one-to-many, e.g., an order is always for a single customer, but a customer can have many orders. In addition, objects may have life cycles, which themselves can be considered as processes. Figure 1 shows three life cycles of objects in the retail shop. A product may be temporarily unavailable, while customers may be blocked by the shop, disallowing them to order products. These life cycles are inherently intertwined. For instance, customers should not be allowed to order products that are unavailable. Similarly, blocked customers should not be able to create new orders.

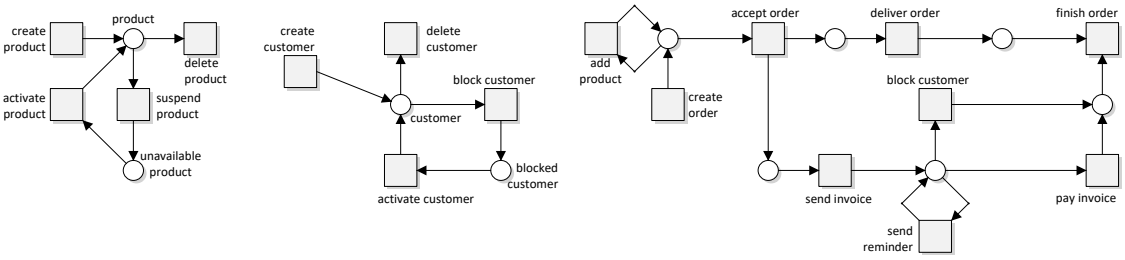


Figure 1: The life cycles of products, customers and orders in the retail shop.

Several approaches have been studied to model and analyze models that combine objects and processes. For example, data-aware Proclets [11] allow describing the behavior of individual artifacts and their interactions. Another approach is followed in ν-PN [12], in which a token can carry a single identifier [13]. In this formalism, markings map each place to a bag of identifiers, indicating how many tokens in each place carry the same identifier. These identifiers can be used to reference entities in an information model. However, referencing a fact composed of multiple entities is not possible in ν-PNs. In this paper, we study *typed Petri nets with identifiers* (t-PNIDs), which build upon ν-PNs [12] by extending tokens to carry vectors of identifiers [6, 7]. Vectors, represented by sequences, have the advantage that a single token can refer to multiple objects or entities that compose (a part of) a *fact*, such as an order is for a specific customer. Identifiers are typed, i.e., the countable, infinite set of identifiers is partitioned into a set of types, such that each type contains a countable, infinite set of identifiers. Identifier types should not overlap, i.e., each identifier has a unique type. Variables can take values of identifiers, and, thus, are typed as well and can only refer to identifiers of the associated type. For example, the product, customer and order objects from the retail shop example make three object types.

Definition 3.1. (Identifier Types)
Let $\mathcal{I}$, Λ, and $\mathcal{V}$ denote countable, infinite sets of identifiers, type labels, and variables, respectively.

We define:

- the *domain assignment* function $I : \Lambda \to \wp(\mathcal{I})$, such that $I(\lambda_1)$ is an infinite set, and $I(\lambda_1) \cap I(\lambda_2) \neq \emptyset$ implies $\lambda_1 = \lambda_2$ for all $\lambda_1, \lambda_2 \in \Lambda$;
- the *id typing* function $type_I : \mathcal{I} \to \Lambda$ s.t. if $type_I(\mathtt{id}) = \lambda$, then $\mathtt{id} \in I(\lambda)$;
- a *variable typing* function $type_{\mathcal{V}} : \mathcal{V} \to \Lambda$, prescribing that $x \in \mathcal{V}$ can be substituted only by values from $I(type_{\mathcal{V}}(x))$.

When clear from the context, we omit the subscripts of $type$.

For ease of presentation, we assume the natural extension of the above typing functions to the cases of sets and vectors, for example, $type_{\mathcal{V}}(x_1 \cdots x_n) = type_{\mathcal{V}}(x_1) \cdots type_{\mathcal{V}}(x_n)$.

In a t-PNID, each place is annotated with a *place type*, which is a vector of types, indicating types of identifier tokens the place can carry. A place with the empty place type, represented by the empty vector, is a classical Petri net place carrying indistinguishable (black) tokens. Each arc of a t-PNID is inscribed with a multiset of vectors of variables, such that the types of the variables in the vector coincide with the place types. This approach allows modeling situations where a transition may require multiple tokens with different identifiers from the same place.

Definition 3.2. (Typed Petri nets with identifiers)
A *Typed Petri net with identifiers* (t-PNID) N is a tuple (P, T, F, α, β), where:

- (P, T, F) is a Petri net;
- $\alpha : P \to \Lambda^*$ is the *place typing function*;

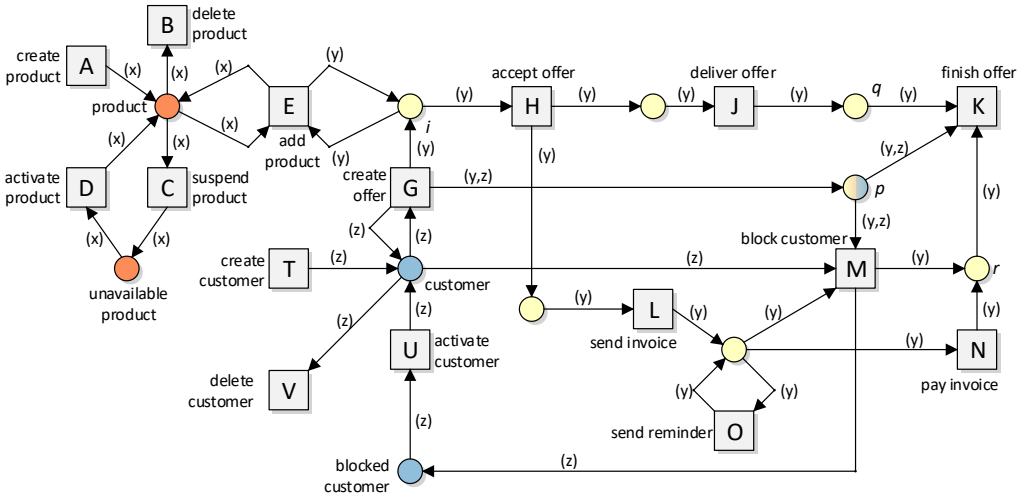


Figure 2: t-PNID N_{rs} for a retail shop that manipulates products, customers and orders. Each place is colored according to its type. Place p carries pairs of identifiers: an order and a customer.

- $\beta : F \to (\mathcal{V}^*)^\oplus$ defines for each flow a multiset of *variable vectors* such that $\alpha(p) = type(\vec{x})$ for any $\vec{x} \in supp(\beta((p,t)))$ and $type(\vec{y}) = \alpha(p')$ for any $\vec{y} \in supp(\beta((t,p')))$ where $t \in T$, $p \in {}^\bullet t, p' \in t^\bullet$;

Figure 2 shows a t-PNID, N_{rs}, of a retail shop. Each place is colored according to its type. The net intertwines the life cycles of Fig. 1 and weakly simulates each of these life cycles. In N_{rs}, places *product* and *unavailable product* are annotated with a vector $\langle product \rangle$, i.e., these places contain tokens that carry only a single identifier of type $product$. Places *customer* and *blocked customer* have type $\langle customer \rangle$. All other places, except for place p, are labeled with type $\langle order \rangle$. Place p maintains the relation between orders and customers, and is typed $\langle order, customer \rangle$, i.e., tokens in this place are identifier vectors of size 2. N_{rs} uses three variables: x for $product$, y for $order$ and z for $customer$.

A marking of a t-PNID N is a configuration of tokens over its places. The set of all possible markings of N is denoted by $\mathbb{M}(N)$. Each token in a place should be of the correct type, i.e., the vector of identifiers carried by a token in a place should match the corresponding place type. All possible vectors of identifiers a place q may carry is defined by the set $\mathtt{C}(q)$.

Definition 3.3. (Marking)
Given a t-PNID $N = (P, T, F, \alpha, \beta)$, and place $p \in P$, its *id set* is $\mathtt{C}(p) = \prod_{1 \le i \le |\alpha(p)|} I(\alpha(p)(i))$. A *marking* is a function $M \in \mathbb{M}(N)$, with $\mathbb{M}(N) = P \to (\mathcal{I}^*)^\oplus$, such that $M(p) \in \mathtt{C}(p)^\oplus$, for each place $p \in P$. The set of identifiers used in m is denoted by $Id(M) = \{\mathtt{id} \mid \exists \vec{\mathtt{id}} \in \mathtt{C}(p), p \in P : \mathtt{id} \in \vec{\mathtt{id}} \wedge M(p)(\vec{\mathtt{id}}) > 0\}$. The pair (N, M) is called a *marked t-PNID*.

To define the semantics of a t-PNID, the variables need to be valuated with identifiers. Variables may be used differently by transitions. In Fig. 2, transition G uses variable y to create an identifier of type $order$. Transition K uses the same variable y to remove identifiers of type $order$ from the marking, as it has no outgoing arcs, and thus only consumes tokens. We, therefore, first introduce some notation to work with variables and types in a t-PNID. Variables used on the input arcs, i.e., variables on arcs from a place to a transition t are called the input variables of t. Similarly, variables on arcs from transition t to a place are called the output variables of t. A variable that only occurs in the set of output variables of a transition, is an *emitting variable*. Similarly, if a variable only appears as an input variable of a transition, it is called a *collecting variable*. As variables are typed, an emitting variable creates a new identifier of a corresponding type upon transition firing, whereas a collecting variable removes the identifier.

Definition 3.4. (Variable sets, emitter and collector transitions, object types)
Given a t-PNID $N = (P, T, F, \alpha, \beta)$, $t \in T$ and $\lambda \in \Lambda$, we define the following sets of variables:

- *input variables* as $In(t) = \bigcup_{\vec{x} \in supp(\beta((p,t))), p \in {}^\bullet t} \bigcup_{x \in \vec{x}} x$;
- *output variables* as $Out(t) = \bigcup_{\vec{x} \in supp(\beta((t,p))), p \in t^\bullet} \bigcup_{x \in \vec{x}} x$;
- *variables* as $Var(t) = In(t) \cup Out(t)$;
- *emitting variables* as $Emit(t) = Out(t) \setminus In(t)$;
- *collecting variables* as $Collect(t) = In(t) \setminus Out(t)$.

Using the above notions, we introduce the sets of:

- *emitting transitions* (or simply referred to as emitters) as $E_N(\lambda) = \{t \mid \exists x \in Emit(t) \wedge type(x) = \lambda\}$;
- *collecting transitions* (or simply referred to as collectors) as $C_N(\lambda) = \{t \mid \exists x \in Collect(t) \wedge type(x) = \lambda\}$.

To properly account for *place types used in* N, we introduce $type_P(N) = \{\vec{\lambda} \mid \exists p \in P : \vec{\lambda} \in \alpha(p)\}$. Similarly, for objects, we introduce the set of *object types in* N $type_\Lambda(N) = \{\lambda \mid \lambda \in \vec{\lambda}, \vec{\lambda} \in type_P(N)\}$.

A firing of a transition requires a *binding* that valuates variables to identifiers. The binding is used to inject new fresh data into the net via variables that emit identifiers. We require bindings to be an injection, i.e., no two variables within a binding may refer to the same identifier. Note that in this definition, the freshness of identifiers is local to the marking, i.e., disappeared identifiers may be reused, as it does not hamper the semantics of the t-PNID. Our semantics allow the use of well-ordered sets of identifiers, such as the natural numbers, as used in [6, 13] to ensure that identifiers are globally new. Here we assume local freshness over global freshness.

Definition 3.5. (Firing rule)
Given a marked t-PNID (N, M) with $N = (P, T, F, \alpha, \beta)$, a *binding* for transition $t \in T$ is an injective function $\psi : \mathcal{V} \to \mathcal{I}$ such that $type(v) = type(\psi(v))$ and $\psi(v) \notin Id(M)$ iff $v \in Emit(t)$. Transition t is *enabled* in (N, M) under binding ψ, denoted by $(N, M)[t, \psi\rangle$ iff $\rho_\psi(\beta(p, t)) \leq M(p)$ for all $p \in {}^\bullet t$. Its firing results in marking M', denoted by $(N, M)[t, \psi\rangle(N, M')$, such that $M'(p) + \rho_\psi(\beta(p, t)) = M(p) + \rho_\psi(\beta(t, p))$.

Again, the firing rule is inductively extended to sequences $\eta \in (T \times (\mathcal{V} \to \mathcal{I}))^*$. A marking M' is *reachable* from M if there exists $\eta \in (T \times (\mathcal{V} \to \mathcal{I}))^*$ s.t. $(N, M)[\eta\rangle(N, M')$. We denote with $\mathcal{R}(N, M)$ the set of all markings reachable from (N, M).

The execution semantics of a t-PNID is defined as an LTS that accounts for all possible executions starting from a given initial marking.

Definition 3.6. (Induced transition system)
Given a marked t-PNID (N, M_0) with $N = (P, T, F, \alpha, \beta)$, its *induced transition system* is $\Gamma_{N,M_0} = (\mathbb{M}(N), (T \times (\mathcal{V} \to \mathcal{I})), M_0, \to)$ with $M \xrightarrow{(t,\psi)} M'$ iff $(N, M)[t, \psi\rangle(N, M')$.

t-PNIDs are a vector-based extension of ν-PNs [12]. In other words, a ν-PN can be translated into a strongly bisimilar t-PNID with a single type, and all place types are of length of at most 1, which follows directly from the definition of the firing rule [12].

Corollary 3.7. For any ν-PN there exists a single-typed t-PNID such that the two nets are strongly rooted bisimilar.

As a result, the decidability of reachability for ν-PNs transfers to t-PNIDs [12].

Proposition 3.8. Reachability is undecidable for t-PNIDs.

4. Correctness criteria for t-PNIDs

Many criteria have been devised for assessing the correctness of systems captured as Petri nets. Traditionally, Petri net-based criteria focus on the correctness of processes the systems can support. Enriching the formalism with ability to capture object manipulation while keeping analyzability is a delicate balancing act.

For t-PNIDs, correctness criteria can be categorized as system-level and object-level. Criteria at the system-level (Section 4.1) focus on traditional Petri net-based criteria to assess the system as a whole, whereas criteria at the object-level (Section 4.2) address the correctness of individual objects represented by identifiers.

4.1. System-level correctness criteria

Liveness is an example of a system-level correctness property. It expresses that any transition is always eventually enabled again. As such, a live system guarantees that its activities cannot eventually become unavailable.

Definition 4.1. (Liveness)
A marked t-PNID (N, M_0) with $N = (P, T, F, \alpha, \beta)$ is *live* iff for every marking $M \in \mathcal{R}(N, M_0)$ and every transition $t \in T$, there exists a marking $M' \in \mathcal{R}(N, M)$ and a binding $\psi : \mathcal{V} \to \mathcal{I}$ such that $M'[t, \psi\rangle$.

Boundedness expresses that the reachability graph of a system is finite, i.e., that the system has finitely many possible states and state transitions. Hence, boundedness is another example of a system-level correctness property. Many systems can support an arbitrary number of simultaneously active objects; they are unbounded by design. Similar to ν-PN, we differentiate between various types of boundedness [14]. Specifically, *boundedness* expresses that the number of tokens in any reachable place does not exceed a given bound. *Width-boundedness* expresses that the modeled system has a bound on the number of simultaneously active objects.

Definition 4.2. (Bounded, width-bounded)
Let (N, M_0) be a marked t-PNID with $N = (P, T, F, \alpha, \beta)$. A place $p \in P$ is called:

- *bounded* if there is $k \in \mathbb{N}$ such that $|M(p)| \leq k$ for all $M \in \mathcal{R}(N, M_0)$;
- *width-bounded* if there is $k \in \mathbb{N}$ such that $|supp(M(p))| \leq k$ for all $M \in \mathcal{R}(N, M_0)$;

If all places in (N, M_0) are (width-) bounded, then (N, M_0) is called (width-) bounded.

As transitions A and T in Fig. 2 have no input places, these transitions are always enabled. Consequently, places *product* and *customer* are not bounded, and thus no place in N_{rs} is bounded. Upon each firing of transition A or T, a new identifier is created. Hence, these places are also not *width-bounded*. In other words, the number of objects in the system represented by N_{rs} is dynamic, without an upper bound.

4.2. Object-level correctness criteria

An object-level property assesses the correctness of individual objects. In t-PNIDs, identifiers can be seen as references to objects: if two tokens carry the same identifier, they refer to the same object. The projection of an identifier on the reachability graph of a marked t-PNID represents the life-cycle of the referenced object. Boundedness of a system implies that the number of states of the reachability graph is finite. *Depth-boundedness* captures this idea for identifiers: in any marking, the number of tokens that refer to a single identifier is bounded. In other words, if a marked t-PNID is depth-bounded, the complete system may still be unbounded, but the life-cycle of each object is finite.

Definition 4.3. (Depth-boundedness)
Let (N, m_0) be a marked t-PNID with $N = (P, T, F, \alpha, \beta)$. A place $p \in P$ is called *depth-bounded* if for each identifier $\mathtt{id} \in \mathcal{I}$ there is $k \in \mathbb{N}$ such that $m(p)(\vec{\mathtt{id}}) \leq k$ for all $m \in \mathcal{R}(N, m_0)$ and $\vec{\mathtt{id}} \in \mathtt{C}(p)$ with $\mathtt{id} \in \vec{\mathtt{id}}$. If all places in P are depth-bounded, (N, m_0) is called depth-bounded.

Depth-boundedness is undecidable for ν-PNs [12] and, thus, also for t-PNIDs.

Proposition 4.4. Depth-boundedness is undecidable for t-PNIDs.

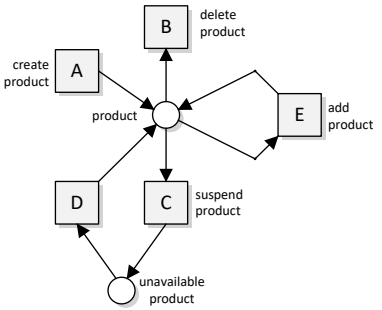
The idea of depth-boundedness is to consider a single identifier in isolation, and study its reachability graph. Intuitively, an object of a given type "enters" the system via an emitter that creates a unique identifier that refers to the object. The identifier remains in the system until the object "leaves" the system by firing a collecting transition (that binds to the identifier and consumes the last token in the net that refers to it). In other words, if a type has emitters and collectors, it has a life-cycle, which can be represented as a process. The process of a type is the model describing all possible paths for the type. It can be derived by taking the projection of the t-PNID on all transitions and places that are "involved" in the type. Notably, the net obtained after the projection is just a regular Petri net.

Definition 4.5. (Type projection)
Let $\lambda \in \Lambda$ be a type. Given a t-PNID $N = (P_N, T_N, F_N, \alpha_N, \beta_N)$, its λ-projection $\pi_\lambda(N) = (P, T, F, W)$ is a Petri net defined by:

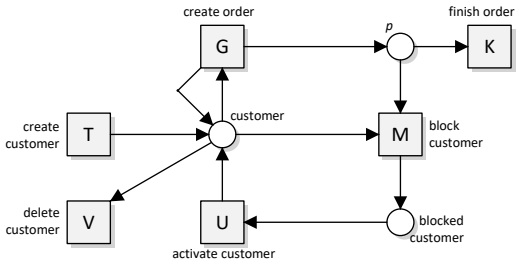
- $P = \{p \in P_N \mid \lambda \in \alpha_N(p)\}$;
- $T = \{t \in T_N \mid \exists p \in P_N : \lambda \in type_{\mathcal{V}}\big(supp(\beta_N((p,t)))\big) \vee \lambda \in type_{\mathcal{V}}\big(supp(\beta_N((t,p)))\big)\}$
- $F = F_N \cap ((P \times T) \cup (T \times P))$;
- $W(f) = |\beta_N(f)|$ for all $f \in F$.

Give a marking $M \in \mathbb{M}(N)$, its λ-projection $\pi_\lambda(M)$ is defined by $\pi_\lambda(M)(p) = |M(p)|$.

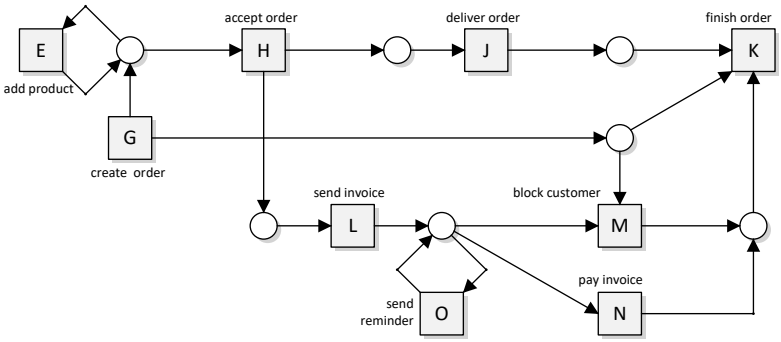
Figure 3 shows the three type projections of N_{rs} from Figure 2. As an emitter of a type creates a new identifier, and a collector removes the created identifier, each type with emitters and collectors can be represented as a transition-bordered WF-net [15]. Instead of a source and a sink place, a transition-bordered WF-net has dedicated transitions that represent the start and finish of a process. A transition-bordered WF-net is sound if its closure is sound [15]. As shown in Fig. 4, the closure



(a) Product projection

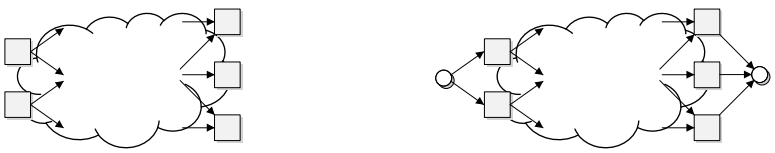


(b) Customer projection



(c) Order projection

Figure 3: Type projections of Figure 2. The customer projection is not sound, as place p is not bounded.



(a) (b)

Figure 4: A transition-bordered WF-Net (a) and its closure (b) [15].

is constructed by creating a new source place so that each emitting transition consumes from it, and a new sink place so that each collecting transition produces in it. In the remainder of this section, we develop this intuition of soundness of type projections into the concept of identifier soundness of t-PNIDs.

Many soundness definitions comprise two properties: proper completion and weak termination. *Proper completion* states that once a marking that has a token in the final marking is reached, it is actually the final marking. For example, for the proper completion to hold in a WF-net, as soon as a

token is produced in the final place, all other places should be empty. Following the idea of transition-bordered WF-nets, identifiers should have a similar property: once a collector consumes one or more identifiers, then no further tokens carrying those identifiers should persist in the marking obtained after the consumption.

Definition 4.6. (Proper type completion)
Given a type $\lambda \in \Lambda$, a marked t-PNID (N, m_0) is called *properly λ-completing* iff for all $t \in C_N(\lambda)$, bindings $\psi : \mathcal{V} \to \mathcal{I}$ and markings $m, m' \in \mathcal{R}(N, m_0)$, if $m[t, \psi\rangle m'$, then for all identifiers $\mathtt{id} \in \text{RNG}(\psi|_{Collect(t)}) \cap Id(m)$ with $type(\mathtt{id}) = \lambda$, it holds that $\mathtt{id} \notin Id(m')$.[2]

Intuitively, from the perspective of a single identifier of type λ, if a t-PNID N that generated it is properly λ-completing, then the points of consumption for this identifier are mutually exclusive (that is, it can be consumed from the net only by one of the collectors from $C_N(\lambda)$).

As an example, consider t-PNID N_{rs} in Fig. 2. For type $customer$, we have $C_{N_{rs}}(customer) = \{K, V\}$. In the current – empty – marking, transition T is enabled with binding $\psi = \{z \mapsto \mathtt{c}\}$, which results in marking m with $m(\textit{customer}) = [\mathtt{c}]$. We can then create an offer by firing G with binding $\psi = \{y \mapsto \mathtt{o}, z \mapsto \mathtt{c}\}$. Next, transitions H, J, L and N can fire, all using the same binding, producing marking m' with $m'(p) = [\mathtt{o}, \mathtt{c}]$, $m'(customer) = [\mathtt{c}]$ and $m'(q) = m'(r) = [\mathtt{c}]$. Hence, transition K is enabled with binding ψ. However, firing K with ψ results in marking m'' with $m''(customer) = [\mathtt{c}]$. Since for the proper type completion on type $customer$ we would like to achieve that all tokens containing $\mathtt{c}$ are removed, N_{rs} is not properly $customer$-completing.

Weak termination signifies that the final marking can be reached from any reachable marking. Translated to identifiers, removing an identifier from a marking should always eventually be possible.

Definition 4.7. (Weak type termination)
Given a type $\lambda \in \Lambda$, a marked t-PNID (N, m_0) is called *weakly λ-terminating* iff for every $m \in \mathcal{R}(N, m_0)$ and identifier $\mathtt{id} \in I(\lambda)$ such that $\mathtt{id} \in Id(m)$, there exists a marking $m' \in \mathcal{R}(N, m)$ with $\mathtt{id} \notin Id(m')$.

Identifier soundness combines the properties of proper type completion and weak type termination: the former ensures that as soon a collector fires for an identifier, the identifier is removed, whereas the latter ensures that it is always eventually possible to remove that identifier.

Definition 4.8. (Identifier soundness)
A marked t-PNID (N, m_0) is *λ-sound* iff it is properly λ-completing and weakly λ-terminating. It is *identifier sound* iff it is λ-sound for every $\lambda \in type_\Lambda(N)$.

Two interesting observations can be made about the identifier soundness property. First, identifier soundness does not imply soundness in the classical sense: any classical net N without types, i.e., $type_\Lambda(N) = \emptyset$, is identifier sound, independently of the properties of N. Second, identifier soundness implies depth-boundedness. In other words, if a marked t-PNID is identifier sound, it cannot accumulate infinitely many tokens carrying the same identifier.

[2] Here, we constrain ψ to objects of type λ that are consumed.

Lemma 4.9. If a t-PNID (N, m_0) is identifier sound, then it is depth-bounded.

Proof:
Suppose that (N, m_0) is identifier sound, but not depth-bounded. Then, at least for one place $p \in P$ and identifier $\vec{\mathtt{id}} \in \mathtt{C}(p)$ of type $\vec{\lambda}$ there exists an infinite sequence of increasing markings m_i, all reachable in (N, m_0), such that $m_i(p)(\vec{\mathtt{id}}) < m_{i+1}(p)(\vec{\mathtt{id}})$. Let $\lambda \in \vec{\lambda}$ and let $\mathtt{id} \in \vec{\mathtt{id}}$ be such that $type(\mathtt{id}) = \lambda$. From the above assumption it follows that there are no such markings m_i and m_{i+1} in the infinite sequence of increasing markings for which it holds that $m_{i+1} \in \mathcal{R}(N, m_i)$, $\mathtt{id} \in m_i(p)$ and $\mathtt{id} \notin m_{i+1}(p)$. Since N is properly type completing, it must be possible to reach from m_i a marking m_i' (via some firing sequence σ) such that $m_i'[t, \psi\rangle m_i''$, for a binding $\psi : \mathcal{V} \to \mathcal{I}$, $m_i'' \in \mathcal{R}(N, m_0)$, $\mathtt{id} \notin Id(m_i'')$ and $t \in C_N(\lambda)$. Since marking m_{i+1} contains at least one more $\mathtt{id}$, then for the same $t \in C_N(\lambda)$ we cannot apply the same reasoning from above. Specifically, we can reach a marking m_{i+1}' from m_{i+1} using the same firing sequence η and, although it still holds that $m_{i+1}'[t, \psi\rangle m_{i+1}''$ (and m_{i+1}'' differs from m_i'' by having one extra $\mathtt{id}$ in p), we have that $\mathtt{id} \in Id(m_i'')$. This contradicts the proper type completion. Hence, (N, m_0) is depth-bounded. □

As identifier soundness relies on reachability, it is undecidable. This also naturally follows from the fact that all non-trivial decision problems are undecidable for Petri nets in which tokens carry pairs of data values (taken from unordered domains) and in which element-wise equality comparisons are allowed over such pairs in transition guards [16].

Theorem 4.10. Identifier soundness is undecidable for t-PNIDs.

Proof:
We prove this result by reduction from the reachability problem for a 2-counter Minsky machine by following ideas of the proof of Theorem 4 in [17].

A 2-counter Minsky machine with two non-negative counters c_1 and c_2 is a finite sequence of numbered instructions $1 : \mathtt{ins}_1, \ldots, n : \mathtt{ins}_n$, where $\mathtt{ins}_n = \mathsf{HALT}$ and for every $1 \leq i < n$ we have that $\mathtt{ins}_i$ has one of the following forms:

- $\mathtt{inc}\ c_j;\ \mathtt{goto}\ k$
- $\mathtt{if}\ c_j = 0\ \mathtt{then\ goto}\ k\ \mathtt{else}\ (\mathtt{dec}\ c_j;\ \mathtt{goto}\ l)$

Here, $j \in \{1, 2\}$ and $1 \leq k, l \leq n$, and $\mathtt{inc}$ (resp., $\mathtt{dec}$) is an operation used to increment (resp., decrement) the content of counter c_j. It is well-known that, for a Minksy 2-counter machine that starts with both counters set to 0, checking whether it eventually reaches the instruction HALT is undecidable.

We then largely rely on the encoding of Minksy 2-counter machines presented in [17]. In a nutshell, that encoding shows how so called OA-nets (we rely on them in the proof of Proposition 4.16) can simulate an arbitrary n-counter Minsky machine. Borrowing an idea from [18], each counter is encoded using a "ring gadget", where counter value m is represented via sets of $m + 1$ linked pairs of identifiers $S = \{(a_1, a_2), (a_2, a_3), \ldots, (a_{m+1}, a_1)\}$ such that an identifier appears exactly twice in S. At the level of the net marking, there always must be only one ring. For more detail on this approach we refer to [17].

Without loss of generality, we assume that the machine halts only when both c_1 and c_2 are zero. Notice that an arbitrary 2-counter machine can be transformed into a corresponding machine that only halts with counter zero by appending, at the end of the original machine, a final set of instructions that decrements both counters, finally halting when they both test to zero.

Using the t-PNID components from Figure 5, we can construct a t-PNID faithfully simulating a 2-counter Minsky machine. Counter operations are defined as in [17]. The t-PNID has one special object type λ, which works as follows:

- a new instance for λ can be only created when a black token is contained in the distinguished *init* place;
- the emission of an object for λ consumes the black token from the *init* place, and inserts it in the place p_{q_0} that corresponds to the first instruction of the 2-counter machine;
- when such a 2-counter machine halts, such a black token is finally transferred into the place p_{q_n}, which in turn enables the last collector transition for λ.[3]

This implies that the t-PNID is λ-sound if and only if the 2-counter machine halts. □

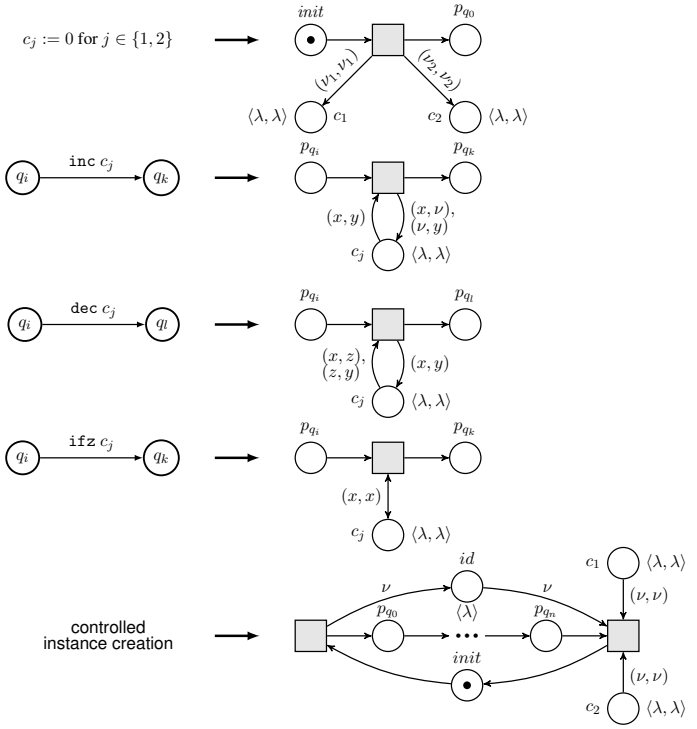
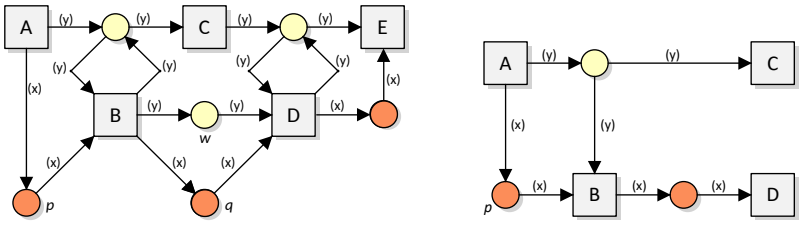


Figure 5: Simulation of a Minksy 2-counter machine via t-PNIDs. Here, q_i, q_k and q_l correspond to control states of the machine.

[3]Notice that transitions of net components simulating instructions with `dec`, according to Definition 3.4, are also collectors for λ.

The above theorem shows that the identifier soundness is already undecidable for nets carrying identifier tuples of size 2. One may wonder whether the same result holds for t-PNIDs with singleton identifiers only. To obtain this result one could, for example, study how the identifier soundness in this particular case relates to the notion of dynamic soundness – an undecidable property of ν-Petri nets studied in [19].



(a) N_1 (b) N_2

Figure 6: Two t-PNIDs. Net N_1 is identifier sound, whereas net N_2 is not identifier sound.

The underlying idea of identifier soundness is that each type projection should behave well, i.e., each type projection should be sound. Consider in t-PNID N_{rs} of Fig. 2 and its $customer$-typed projection from Fig. 3b. The life cycle starts with transition T. Transitions K and V are two transitions that may remove the last reference to a $customer$. Soundness of a transition-bordered WF-net would require that firing transition K or transition V would result in the final marking. However, this is not necessarily the case. Consider the firing of transitions T, G and K. Then, the token in place *customer* remains, while the final transition K already fired. Hence, the *customer* life cycle is not sound. This raises the question whether we may conclude from this observation that N_{rs} is not identifier sound. Unfortunately, identifier soundness is not compositional, i.e., identifier soundness does not imply soundness of the type projections, and vice versa. Consider the example t-PNIDs in Fig. 6. The first net, N_1, is identifier sound. However, taking the $type_{\mathcal{V}}(y)$ projection of N_1 results in a bordered transition WF-net that is unsound: if transition D fires fewer times than transition B, tokens will remain in place w. The reverse is false as well. Consider net N_2 in Fig. 6. Each of the two projections are sound. However, the resulting net reaches a deadlock after firing transitions A and C as a token generated by A remains in place p. Hence, N_2 is not weakly $type_{\mathcal{V}}(x)$-completing.

Theorem 4.11. Let $\lambda \in \Lambda$ be a type, and let (N, m_0) be a marked t-PNID. Then:

1. identifier soundness of (N, m_0) does not imply soundness of $(\pi_\lambda(N), \pi^N_\lambda(m))$, and
2. soundness of $(\pi_\lambda(N), \pi^N_\lambda(m))$ does not imply that (N, m_0) is identifier sound.

Proof:
We prove both statements by contradiction. For the first statement, consider t-PNID N_1 depicted in Fig. 6a. Though N_1 is identifier sound, its $type_{\mathcal{V}}(y)$-projection is not sound. Similarly, the $type_{\mathcal{V}}(x)$-projection and $type_{\mathcal{V}}(y)$-projection of N_2, depicted in Fig. 6b, are sound, but N_2 is not identifier sound, as N_2 is not weakly $type_{\mathcal{V}}(x)$-completing. □

Consequently, compositional verification of soundness of each of the projections is not sufficient to conclude anything about identifier soundness of the complete net, and vice versa.

In general, weak bisimulation does not guarantee identifier soundness, as it does not impose any relation on the identifiers in the nets. However, if the bisimulation relation takes into account and preserves identifiers, then identifier soundness is preserved. We formally demonstrate this property below.

Lemma 4.12. (Weak bisimulation preserves proper type completion)
Let (N_1, m_0^1) and (N_2, m_0^2) be two marked t-PNIDs. Let $\lambda \in \Lambda$ be some type such that $C_{N_1}(\lambda) = C_{N_2}(\lambda)$, and let $Q \subseteq \mathbb{M}(N_1) \times \mathbb{M}(N_2)$ be a relation such that $I(\lambda) \cap Id(m_1) = I(\lambda) \cap Id(m_2)$ for all $(m_1, m_2) \in Q$ and $\Gamma_{N_1,m_0^1} \approx_Q \Gamma_{N_2,m_0^2}$. Then N_1 is properly λ-completing iff N_2 is properly λ-completing.

Proof:
($\Rightarrow$) Suppose N_1 is properly λ-completing. We need to show that N_2 is properly λ-completing. Let $t \in C_{N_2}(\lambda)$ be a transition, let $\psi : \mathcal{V} \to \mathcal{I}$ be a binding and let $m_2, m_2' \in \mathcal{R}(N_2, m_0^2)$, such that $(N_2, m_2)[t, \psi\rangle(N_2, m_2')$. Let $\mathtt{id} \in \text{RNG}(\psi|_{Collect(t)}) \cap Id(m_2)$ with $type(\mathtt{id}) = \lambda$. As Q is a weak bisimulation relation, some markings $m_1, m_1' \in \mathbb{M}(N_1)$ exist such that $(m_1, m_2) \in Q$ and $(N_1, m_1)[t, \psi\rangle(N_1, m_1')$. Then, by the definition of Q, it must be that $\mathtt{id} \in \text{RNG}(\psi|_{Collect(t)}) \cap Id(m_1)$. As $C_{N_1}(\lambda) = C_{N_2}(\lambda)$ and N_1 is properly λ-completing, $\mathtt{id} \notin Id(m_1')$. Since Q is a weak bisimulation relation, $(m_1', m_2') \in Q$. Thus, $\mathtt{id} \notin Id(m_2')$. Hence, N_2 is properly λ-completing.

($\Leftarrow$) Follows from the commutativity of weak bisimulation. □

Lemma 4.13. (Weak bisimulation preserves weak type termination)
Let (N_1, m_0^1) and (N_2, m_0^2) be two marked t-PNIDs. Let $\lambda \in \Lambda$ be some type, and let $Q \subseteq \mathbb{M}(N_1) \times \mathbb{M}(N_2)$ be a relation such that $I(\lambda) \cap Id(m_1) = I(\lambda) \cap Id(m_2)$ for all $(m_1, m_2) \in Q$ and $\Gamma_{N_1,m_0^1} \approx_Q \Gamma_{N_2,m_0^2}$. Then N_1 is weakly λ-terminating iff N_2 is weakly λ-terminating.

Proof:
($\Rightarrow$) Suppose N_1 is weakly λ-terminating. We need to show that N_2 is weakly λ-terminating. Let $m_2 \in \mathcal{R}(N_2, m_0^2)$ be some reachable marking and $\mathtt{id} \in I(\lambda)$ such that $\mathtt{id} \in Id(m_2)$. As Q is a weak bisimulation relation, some marking $m_1 \in \mathbb{M}(N_1)$ exists with $(m_1, m_2) \in Q$. Then, by the definition of Q, $\mathtt{id} \in Id(m_1)$. As N_1 is weakly λ-terminating, a marking $m_1' \in \mathcal{R}(N_1, m_1)$ and firing sequence η exist such that $(N_1, m_1)[\eta\rangle(N_1, m_1')$ and $\mathtt{id} \notin Id(m_1')$. As Q is a weak bisimulation relation, a marking $m_2' \in \mathbb{M}(N_2)$ exists such that $(m_1', m_2') \in Q$ and $(N_2, m_2)[\eta\rangle(N_2, m_2')$. As $I(\lambda) \cap Id(m_1') = I(\lambda) \cap Id(m_2')$, we have that $\mathtt{id} \notin Id(m_2')$, which proves the statement.

($\Leftarrow$) Follows from the commutativity of weak bisimulation. □

The two above lemmas are combined together to prove the following result.

Theorem 4.14. (Weak bisimulation preserves λ-soundness)
Let (N_1, m_0^1) and (N_2, m_0^2) be two marked t-PNIDs. Let $\lambda \in \Lambda$ be some type such that $C_{N_1}(\lambda) = C_{N_2}(\lambda)$, and let $Q \subseteq \mathbb{M}(N_1) \times \mathbb{M}(N_2)$ be a relation such that $I(\lambda) \cap Id(m_1) = I(\lambda) \cap Id(m_2)$ for all $(m_1, m_2) \in Q$ and $\Gamma_{N_1,m_0^1} \approx_Q \Gamma_{N_2,m_0^2}$. Then N_1 is λ sound iff N_2 is λ sound.

Proof:
Follows directly from Lm 4.12 and Lm 4.13. □

4.3. Towards verification of logical criteria

We now describe how existing results on the verification of safety and temporal properties over variants of Petri nets [20, 17] and transition systems operating over relational structures [21, 22] can be lifted to the case of *bounded* t-PNIDs. Boundedness is a sufficient requirement to make the verification of such properties decidable.

We start by considering safety checking of t-PNIDs, considering the recent results presented in [17]. A safety property is a property that must hold globally, that is, in every marking of the net. Such a property is usually checked by formulating its unsafety dual and verifying whether a marking satisfying that unsafety property is reachable.

Definition 4.15. (Unsafety property)
An *unsafety property* over a t-PNID N is a formula $\exists y_1, \ldots, y_k.\psi(y_1, \ldots, y_k)$, where ψ is defined by the following grammar:

$$\psi ::= p(x_1 \cdots x_n) \geq c \,|\, x = y \,|x \neq y \,|\, p \geq c \,|\, \psi \wedge \psi,$$

Here: *(i)* p is a place name from N, *(ii)* $x_i \in \mathcal{V}$ (for every $1 \leq i \leq n$), *(iii)* $x, y \in \mathcal{V} \cup \mathcal{I}$, *(iv)* $p \geq c$ and $p(x_1 \cdots x_n) \geq c$ are atomic predicates defined over place markings with $c \in \mathbb{N}$.

Given a marked t-PNID (N, m_0), each atomic predicate is interpreted on all possible markings covering those from $\mathcal{R}(N, m_0)$. Like that, $p \geq c$ specifies that in place p there are at least c tokens, whereas $p(x_1 \cdots x_n) \geq c$ indicates that in place p there are at least c tokens carrying an identifier vector that can valuate $x_1 \cdots x_n$. We use elements from $x_1, \ldots, x_n$ as a filter selecting matching tokens in p, and use the variables from the same sequence to inspect different places by creating implicit joins between tokens stored therein. As it has been established in [17], such unsafety properties can be used for expressing object-aware *coverability* properties of t-PNIDs.

For example, as a property we may write that $\exists z, y.created\_offer(z, y) \geq 1 \wedge customer(y) \geq 1$ captures the (undesired) situation in which an offer has been made to customer z, but that customer is still available for receiving other offers.

The verification problem for checking unsafety properties is specified as follows: given an unsafety property ψ, a marked t-PNID (N, m_0) is *unsafe* w.r.t. ψ if (N, m_0) can reach a marking in which ψ holds. If this is not the case, then we say that (N, m_0) is *safe* w.r.t. ψ. We show below that such verification problem is actually decidable.

Proposition 4.16. Verification of unsafety properties over bounded, marked t-PNIDs is decidable.

Proof:
To prove this statement, we introduce the class of OA-nets [17] and establish their relation to t-PNIDs. To do so, we provide a modified version of Definition 1 from [17]. Essentially, an OA-net is a tuple $(P, T, F_{in}, F_{out}, \texttt{color}, \texttt{guard})$, where:

- P and T are finite sets of places and transitions, s.t. $P \cap T = \emptyset$;
- $\texttt{color} : P \to \Lambda^*$ is a place typing function;
- $F_{in} : P \times T \to \Omega_{\mathcal{V}}^{\oplus}$ is an input flow s.t. $type_{\mathcal{V}}(F_{in}(p,t)) = \texttt{color}(p)$ for every $(p,t) \in P \times T$;[4][5]
- $F_{out} : T \times P \to \Omega_{\mathcal{X}}^{\oplus}$ is an output flow s.t. $type_{\mathcal{V}}(F_{out}(t,p)) = \texttt{color}(p)$ for every $(t,p) \in T \times P$ and $\mathcal{X} = \mathcal{V} \uplus \hat{\mathcal{V}}$, where $\hat{\mathcal{V}}$ is the countably infinite set of fresh variables (i.e., variables used to provide fresh inputs only);[6]
- $\texttt{guard} : T \not\to \Phi$ is a partial guard assignment function, s.t. Φ is a set of conditions $\phi ::= y_1 = y_2 \,|\, y_1 \neq y_2 \,|\, \phi \wedge \phi$, where $y_i \in \mathcal{V} \cup \mathcal{I}$, and for each $\phi = \texttt{guard}(t)$ and $t \in T$ it holds that $Var(\phi) \subseteq In(t)$.[7]

It is easy to see that a t-PNID is an OA-net without guards. Moreover, using the notions from Definition 3.4, we get that $Emit(t) \subset \hat{\mathcal{V}}$, for each $t \in T$.

Given the relation between t-PNIDs and OA-nets, the proof of the decidability follows immediately from Theorem 3 in [17]. □

One may wonder whether it is possible to go beyond safety and check other properties expressible on top of t-PNIDs using more sophisticated temporal logics. We answer to this question affirmatively, by proving that bounded t-PNIDs induce transition systems that enjoy the so-called *genericity* property [21]. Such property, combined with t-PNID boundedness (which corresponds to the notion of state-boundedness used in [21]) guarantees decidability of model checking for sophisticated variants of first-order temporal logics [23, 21, 22].

In generic transition systems, the behaviour does not depend on the actual data present in the states, but only on how they relate to each other. This essentially reconstructs the well-known notion of genericity in databases, which expresses that isomorphic databases return the same answers to the same query, modulo renaming of individuals [24].

We lift the notion of genericity to the case of transition systems induced by t-PNIDs. To proceed, we first need to define a suitable notion of isomorphism between two markings of a net.

Definition 4.17. (Marking isomorphism)
Given a t-PNID N and two markings $m_1, m_2 \in \mathbb{M}(N)$, we say that m_1 and m_2 are *isomorphic*, written $m_1 \sim_h m_2$, if there exits a bijection (called *isomorphism*) $h : Id(m_1) \to Id(m_2)$ such that for every $p \in P$, it holds that $(\texttt{id}_1 \cdots \texttt{id}_n)^k \in m_1$ iff $(h(\texttt{id}_1) \cdots h(\texttt{id}_n))^k \in m_2$, for every $k \in \mathbb{N}$.

Intuitively, two markings are called isomorphic if they have the same amount of tokens and tokens correspond to each other modulo consistent renaming of identifiers.

[4] We denote by Ω_A the set of all possible tuples of variables and identifiers over a set A.

[5] Without loss of generality, we assume that $type_{\mathcal{V}}$ naturally extends to cartesian products.

[6] The original definition from [17] also allows for constants from $\mathcal{I}$ to appear in the output flow vectors. However, for simplicity's sake, we removed it here form the definition of F_{out}.

[7] Here, $Var(\phi)$ provides the set of all variables in ϕ.

Definition 4.18. (Generic transition system)
Let $\Gamma = (S, A, s_0, \rightarrow)$ be the transition system induced by some t-PNID. Then Γ is *generic* if for every markings $m_1, m_1', m_2 \in S$ and every bijection $h : \mathcal{I} \rightarrow \mathcal{I}$, if $m_1 \sim_h m_2$ and $m_1[t, \psi\rangle m_1'$ (for some $t \in T$ and binding $\psi : \mathcal{V} \rightarrow \mathcal{I}$), then there exists $m_2' \in S$ and $\psi' : \mathcal{V} \rightarrow \mathcal{I}$ such that $m_2[t, \psi'\rangle m_2'$, $m_1' \sim_h m_2'$ and $\psi(v) = h(\psi'(v))$, for every $v \in \mathcal{V}$.

As one can see from the definition, genericity requires that if two marking are isomorphic, then they induce the same transitions modulo isomorphism (i.e., the transition names are the same, and the variable assignments are equivalent modulo renaming). This implies that they induce isomorphic successors.

Remark 4.19. Let (N, m_0) be a marked t-PNID. Then its induced transition system Γ_{N,m_0} is generic.

The above result can be easily shown by considering the transition system construction described in Definition 3.6, by considering isomorphism between its markings given by a simple renaming function and checking the conditions of Definition 4.18.

As it has been demonstrated in [21, 22], model checking of sophisticated first-order variants of μ-calculus and LTL becomes decidable for so-called state-bounded generic transition systems. Since the transition systems induced by t-PNIDs are also generic, and boundedness of a t-PNID correspond to state-boundedness of its induced transition system, we directly obtain decidability of model checking for the same logics considered there, extended in our case over atomic predicates of the form $p \odot c$ and $p(x_1 \cdots x_n) \odot c$, where $\odot \in \{<, >, =, \leq, \geq\}$, in the style of similar logics introduced in [20]. We shall refer to such logics as $\mu\mathcal{L}$-FO^{PNID} and LTL-FO_p^{PNID}, but in this work we omit their definition.

Theorem 4.20. Model checking of $\mu\mathcal{L}$-FO^{PNID} and LTL-FO_p^{PNID} formulae is decidable for bounded, marked t-PNIDs.

Whereas the boundedness condition may appear restrictive at the first sight, we recall that according to Definition 4.2, a t-PNID is k-bounded if every marking reachable from the initial one does not assign more than k tokens to every place of the net. This however does not impede the net at hand from reaching infinitely many states as tokens may, along its run, carry infinitely many distinct objects. Notice also that this condition is less restrictive than identifier boundedness (which essentially forces the identifier domain to be finite) made use of in [6]. In Section 6 we discuss a class of t-PNIDs for which boundedness still allows to explore lifecycles of potentially infinitely many objects.

5. Correctness by construction

As shown in the previous section, identifier soundness is undecidable. However, we are still interested in ensuring correctness criteria over the modeled system. In this section, we propose a structural approach to taming the undecidability and study sub-classes of t-PNIDs that are identifier sound by construction.

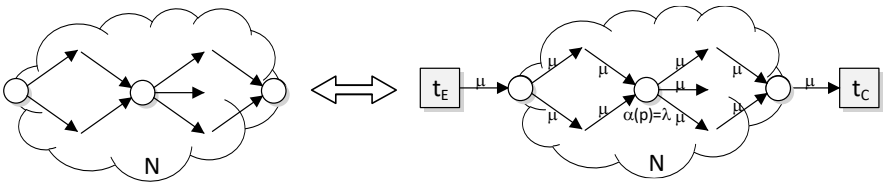


Figure 7: EC-closure of a WF-net N.

5.1. EC-closed workflow nets

WF-nets are widely used to model business processes. The initial place of the WF-net signifies the start of a *case*, the final place represents the goal state, i.e., the process case completion. A firing sequence from initial state to final state represents the activities that are performed for a single case. Thus, a WF-net describes all possible sequences of a single case. Process engines, like Yasper [25] simulate the execution of multiple cases in parallel by coloring the tokens with the case identifier (a similar idea is used for resource-constrained WF-net variants of ν-PNs in [26]). In other words, they label each place with a case type, and inscribe each arc with a variable. To execute it, the WF-net is closed with an emitter and a collector, as shown in Figure 7. We generalize this idea to any place label, i.e., any finite sequence of types may be used to represent a case. We use the technical results obtained in this section further on in Section 5.3.

Definition 5.1. (EC-Closure)
Given a WF-net N, place type $\vec{\lambda} \in \Lambda^*$ and a variable vector $\vec{v} \in \mathcal{V}^*$ such that $type_{\mathcal{V}}(\vec{v}) = \vec{\lambda}$, its *EC-closure* is a t-PNID $\mathcal{W}(N, \vec{\lambda}, \vec{v}) = (P_N, T_N \cup \{t_E, t_C\}, F_N \cup \{(t_E, in), (out, t_C)\}, \alpha, \beta)$, with:

- $\alpha(p) = \vec{\lambda}$ for all places $p \in P_N$;
- $\beta(f) = \vec{v}^{\,W(f)}$ for all flows $f \in F_N$, and $\beta((t_e, in)) = \beta((out, t_c)) = [\,\vec{v}\,]$.

The EC-closure of a WF-net describes all cases that run simultaneously at any given time. In other words, any reachable marking of the EC-closure is the "sum" of all simultaneous cases. Lemma 5.2 formalizes this idea by establishing weak bisimulation between the projection on a single case and the original net.

Lemma 5.2. (Weak bisimulation for each identifier)
Let N be a WF-net, $\vec{\lambda} \in \Lambda^*$ be a place type and $\vec{v} \in \mathcal{V}^*$ be a variable vector s.t. $type_{\mathcal{V}}(\vec{v}) = \vec{\lambda}$. Then, for any $\vec{\mathtt{id}} \in \mathcal{I}^{|\vec{\lambda}|}$, $\rho_r(\Gamma_{\mathcal{W}(N,\vec{\lambda},\vec{v}),\emptyset}) \approx \Gamma_{N,[in]}$, where r in renaming ρ_r is such that $r(t, \psi) = t$, if $\psi(\vec{v}) = \vec{\mathtt{id}}$, and $r((t, \psi)) = \tau$, otherwise.

Proof:
Let $N' = \mathcal{W}(N, \vec{\lambda}, \vec{v})$. Define $R = \{(M, m) \mid \forall p \in P : M(p)(\vec{\mathtt{id}}) = m(p)\}$. We need to show that R is a weak bisimulation.
($\Rightarrow$) Let M, M' and m be such markings that $(M, m) \in R$ and $(N', M)[t, \psi\rangle(N', M')$, with $t \in T$ and $\psi : \mathcal{V} \to \mathcal{I}$. By Definition 5.1, $\psi(\vec{v}) = \vec{\mathtt{u}}$, for some $\vec{\mathtt{u}} \in \mathcal{I}^{|\vec{\lambda}|}$. From the firing rule, we obtain

$M'(p) + [\vec{\mathtt{u}}^{W((p,t))}] = M(p) + [\vec{\mathtt{u}}^{W((t,p))}]$, for any $p \in P$. If $\vec{\mathtt{u}} \neq \vec{\mathtt{id}}$, then $r(t,\psi) = \tau$, and $(M', m) \in R$. If $\vec{\mathtt{u}} = \vec{\mathtt{id}}$, there exists such marking m' that $m[t\rangle m'$ (since $m(p) = M(p)(\vec{\mathtt{id}})$ and thus $m(p) \geq W((p,t))$) and $m'(p) + W((p,t)) = M(p)(\vec{\mathtt{id}}) + W((t,p))$. Then, by construction, $m'(p) = M'(p)(\vec{\mathtt{id}})$ and $(M', m') \in R$.

($\Leftarrow$) Let M, m, and m' be markings that $(M, m) \in R$ and $(N, m)[t\rangle(N, m')$ with $t \in T$. We choose binding ψ such that $\psi(\vec{v}) = \vec{\mathtt{id}}$. Then $\rho_\psi(\beta(p,t)) = [\vec{\mathtt{id}}^{W((p,t))}] \leq M(p)$, since $W((p,t)) \leq m(p) = M(p)(\vec{\mathtt{id}})$. Thus, a marking M' exists such that $(N', M)[t, \psi\rangle(N', M')$. Then $M'(p) + [\vec{\mathtt{id}}^{W((p,t))}] = M(p) + [\vec{\mathtt{id}}^{W((t,p))}]$. Hence, $M'(p)(id) = m'(p)$ and thus $(M', m') \in R$. □

A natural consequence of this weak bisimulation result is that any EC-closure of a WF-net is identifier sound if the underlying WF-net is sound.

Theorem 5.3. Given a WF-Net N, if N is sound, then $\mathcal{W}(N, \vec{\lambda}, \vec{v})$ is identifier sound and live, for any place type $\vec{\lambda} \in \Lambda^*$ and variable vector $\vec{v} \in \mathcal{V}^*$ with $type(\vec{v}) = \vec{\lambda}$.

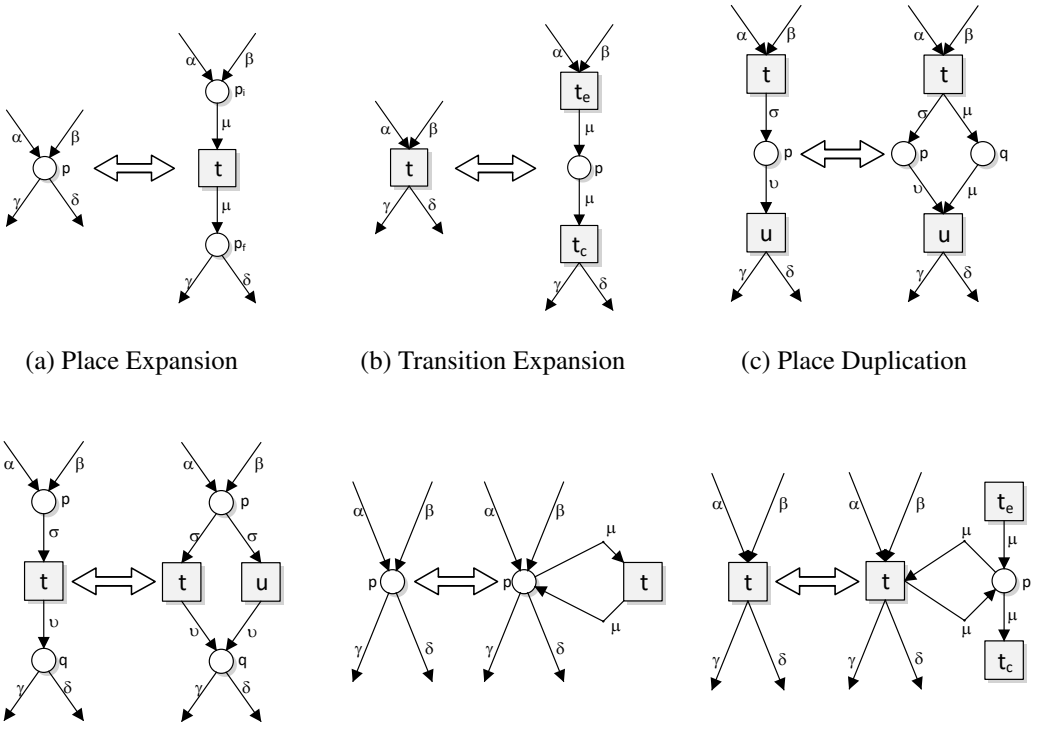
Proof:
Let $N' = \mathcal{W}(N, \vec{\lambda}, \vec{v}) = (P, T, F, \alpha, \beta)$. By definition of $\mathcal{W}$, $Collect(t) = \emptyset$ for any transition $t \in T \setminus \{t_C\}$. Hence, only transition t_C can remove identifiers, and thus, by construction, $\mathcal{W}$ is properly type completing on all $\lambda \in \vec{\lambda}$.

Next, we need to show that N' is weakly type terminating for all types $\lambda \in \vec{\lambda}$. Let $M \in \mathcal{R}(N', \emptyset)$, with firing sequence $\eta \in (T \times (\mathcal{V} \to \mathcal{I}))^*$, i.e., $(N', \emptyset)[\eta\rangle(N', m)$. Let $\vec{\mathtt{id}} \in \mathtt{C}(p)$ such that $M(p)(\vec{\mathtt{id}}) > 0$ for some $p \in P$. We then construct a sequence ω by stripping the bindings from η s.t. it contains only transitions of T. Using Lemma 5.2, we obtain a marking $m \in \mathcal{R}(N, [in])$ such that $[in][\psi\rangle m$ and $m(p) = M(p)(\vec{\mathtt{id}})$. Since N is sound, there exists a firing sequence ω' such that $m[\omega'\rangle[out]$. Again by Lemma 5.2, a firing sequence η' exists such that $M[\eta'\rangle M'$ and $(M', [out]) \in \mathcal{R}(\mathcal{W}, \emptyset)|_{\vec{\mathtt{id}}}$, where $\mathcal{R}(\mathcal{W}, \emptyset)|_{\vec{\mathtt{id}}}$ is the set of all reachable markings containing $\vec{\mathtt{id}}$. Hence, if $M'(p)(\vec{\mathtt{id}}) > 0$, then $p = out$. Thus, transition t_C is enabled with some binding ψ such that $\psi(\vec{v}) = \vec{\mathtt{id}}$, and a marking M'' exists such that $M'[t_c, \psi\rangle M''$, which removes all identifiers in $\vec{\mathtt{id}}$ from M'. Hence, N' is identifier sound.

As transition t_e is always enabled and N is quasi live, $\mathcal{W}(N, \vec{\lambda}, \vec{v})$ is live. □

5.2. Typed Jackson nets

A well-studied class of processes that guarantee soundness are block-structured nets. Examples include Process Trees [27], Refined Process Structure Trees [28] and Jackson Nets [29]. Each of the techniques have a set of rules in common from which a class of nets can be constructed that guarantees properties like soundness. In this section, we introduce Typed Jackson Nets (t-JNs), extending the ideas of Jackson Nets [29, 15] to t-PNIDs, that guarantee both identifier soundness and liveness. The six reduction rules presented by Murata in [30] form the basis of this class of nets. The rules for t-JNs are depicted in Figure 8.



(a) Place Expansion (b) Transition Expansion (c) Place Duplication

(d) Transition Duplication (e) Self-loop Addition (f) Identifier Creation

Figure 8: Construction rules of the typed Jackson Nets.

5.2.1. Place Expansion

The first rule is based on *fusion of a series of places*. As shown in Figure 8a, a single place p is replaced by two places p_i and p_f that are connected via transition t. All transitions that originally produced in p, produce in p_i in the place expansion, and similarly, the transitions that consumed from place p, now consume from place p_f. In fact, transition t can be seen as a transfer transition: it needs to move tokens from place p_i to place p_f, before the original process can continue. This is also reflected in the labeling of the places: both places have the same place type, and the input and output arc of transition t are inscribed with the same variable vector $[\,\vec{\mu}\,]$ that matches the type of place p.

Definition 5.4. (Place expansion)
Let (N, m) be a marked t-PNID with $N = (P, T, F, \alpha, \beta)$, $p \in P$ be a place and $\vec{\mu} \in \mathcal{V}^*$ be a variable vector s.t. $type_{\mathcal{V}}(\vec{\mu}) = \alpha(p)$. The *place expanded t-PNID* is defined by the relation $R_{p,\vec{\mu}}(N, m) = ((P', T', F', \alpha', \beta'), m')$, where:

- $P' = (P \setminus \{p\}) \cup \{p_i, p_f\}$ with $p_i, p_f \notin P$; and $T' = T \cup \{t\}$ with $t \notin T$;
- $F' = (F \setminus ((\{p\} \times p^\bullet) \cup ({}^\bullet p \times \{p\})) \cup ({}^\bullet p \times \{p_i\}) \cup \{(p_i, t), (t, p_f)\} \cup (\{p_f\} \times p^\bullet)$;
- $\alpha'(q) = \alpha(p)$, if $q \in \{p_i, p_f\}$, and $\alpha'(q) = \alpha(q)$, otherwise.

- $\beta'(f) = [\vec{\mu}]$, if $f \in \{(p_i, t), (t, p_f)\}$, $\beta'((u, p_i)) = \beta((u, p))$, if $u \in {}^\bullet p$, $\beta'((p_f, u)) = \beta((p, u))$, if $u \in p^\bullet$, and $\beta'(f) = \beta(f)$, otherwise.
- $m'(q) = m(q)$ for all $q \in P \setminus \{p\}$, $m'(p_f) = 0$, and $m'(p_i) = m(p)$.

Inscription $\vec{\mu}$ cannot alter the vector identifier on the tokens, as the type of $\vec{\mu}$ should correspond to both place types $\alpha(p)$ and $\alpha(q)$. Hence, the transition is enabled with the same bindings as any other transition that consumes a token from place p, modulo variable renaming. As such, transition t only "transfers" tokens from place p_i to place p_f. Hence, as the next lemmas shows, place expansion yields a weakly bisimilar t-PNID and preserves identifier soundness.

Lemma 5.5. Let (N, m_0) be a marked t-PNID with $N = (P, T, F, \alpha, \beta)$, $p \in P$ be a place to expand and $\vec{\mu} \in \mathcal{V}^*$ be a variable vector s.t. $type_{\mathcal{V}}(\vec{\mu}) = \alpha(p)$. Then $\Gamma_{N,m_0} \approx^r \hat{\mathfrak{H}}_{\{t\}}(\Gamma_{R_{p,\vec{\mu}}(N,m_0)})$, with transition t added by $R_{p,\vec{\mu}}$.

Proof:
Let $(N', m'_0) = R_{p,\mu}(N, m_0)$. We define $Q \subseteq \mathbb{M}(N) \times \mathbb{M}(N')$ such that $(m, m') \in Q$ iff $m(q) = m'(q)$ for all places $q \in P \setminus \{p\}$ and $m'(p_i) + m'(p_f) = m(p)$. Then $(m_0, m'_0) \in Q$, hence the relation is rooted.

($\Rightarrow$) Let $(m, m') \in Q$ and $(N, m)[u, \psi\rangle(N, \bar{m})$. We need to show that there exists marking $\bar{m}'$ such that $m' \xrightarrow{(u,\psi)} \bar{m}'$ and $(\bar{m}, \bar{m}') \in Q$.

Suppose $p \notin {}^\bullet u$. Then $m'(q) = m(q)$ and $m(q) \geq \rho_\psi(\beta((p, u)))$ (note that $\rho_\psi(\beta((p, u))) = \rho_\psi(\beta'((p, u)))$). By the firing rule, a marking $\bar{m}'$ exists with $(N, m')[u, \psi\rangle(N', \bar{m}')$, $\bar{m}(q) = \bar{m}'(q)$ for all $q \in P'$. Thus, $(\bar{m}, \bar{m}') \in Q$. Suppose $p \in {}^\bullet u$. Then $\rho_\psi(\beta((p_f, u))) \leq m(p) = m'(p_i) + m'(p_f)$. If $\rho_\psi(\beta(p_f, u))) \leq m'(p_f)$, then transition u is enabled, and a marking $\bar{m}'$ exists with $(N, m')[u, \psi\rangle(N, \bar{m}')$ and $(\bar{m}, \bar{m}') \in Q$.

Otherwise, $\rho_\psi(\beta(p_f, u))) \leq m'(p_i)$. Construct a binding ψ' by letting $\psi'(\mu(i)) = \psi(\beta(p, u)(i))$, for all $1 \leq i \leq |\mu|$. Then, $\rho_{\psi'}(\mu) = \rho_\psi(\beta(p, u))$, and transition t is enabled with binding ψ'. Hence, a marking m'' exists with $(N, m')[t, \psi'\rangle(N, m'')$ and $\rho_\psi(\beta((p', u))) \leq m''(p')$. Then $(m, m'') \in Q$ and t is labeled τ in $\hat{\mathfrak{H}}_{\{t\}}(R_{(p,\vec{\mu})}(N))$. Now, either transition u is enabled, or transition t is again enabled with binding ψ'. In all cases, $m' \xrightarrow{(t,\psi)} \bar{m}'$ and $(m', \bar{m}') \in Q$.

($\Leftarrow$) Let $(m, m') \in Q$ and $(N', m')[u, \psi\rangle(N', \bar{m}')$. We need to show that either a $\bar{m}$ exists such that $(N, m)[u, \psi\rangle(N, \bar{m})$ and $(\bar{m}, \bar{m}') \in Q$ or $u = \tau$ and $(m, \bar{m}') \in Q$.

Suppose $u = t$, i.e., u is labeled τ in $\hat{\mathfrak{H}}_{\{t\}}(R_{(p,\vec{\mu})}(N))$. Then, ${}^\bullet u = p_i$ and $u^\bullet = p_f$. By the firing rule, $\bar{m}'(p_i) + \bar{m}'(p_f) = m(p_i) + m(p_f) = m(p)$. Hence $(m, \bar{m}') \in Q$.

If $u \neq t$, we need to show that there exists marking $\bar{m}$ such that $m \xrightarrow{(u,\psi)} \bar{m}$ and $(\bar{m}, \bar{m}') \in Q$. Let $q \in {}^\bullet u$. If $q \neq p$, then $m(q) \leq \rho_\psi(\beta(p, u))$. If $q = p$, then $m(q) \geq m(p_f)$ and thus, $m(q) \leq \rho_\psi(\beta(p, u))$. Hence, transition u is enabled in m and a marking $\bar{m}$ exists such that $(N, m)[t, \psi'\rangle(N, \bar{m})$. By the firing rule, we have $\bar{m}(p) = m(p) - \rho_\psi(\beta(p, u)) + \rho_\psi(\beta(u, p)) = m'(p_i) + m'(p_f) - \rho_\psi(\beta(p_i, u)) - \rho_\psi(\beta(p_f, u)) + \rho_\psi(\beta(u, p_i)) + \rho_\psi(\beta(u, p_f)) = \bar{m}'(p_i) + \bar{m}'(p_f)$, since $\rho_\psi(\beta(p_i, u)) = \rho_\psi(\beta(u, p_f) = \emptyset$. Hence, $(\bar{m}, \bar{m}') \in Q$, which proves the statement. □

Lemma 5.6. Let (N, m_0) be a marked t-PNID with $N = (P, T, F, \alpha, \beta)$, $p \in P$ be a place to expand and $\vec{\mu} \in \mathcal{V}^*$ be a variable vector s.t. $type_{\mathcal{V}}(\vec{\mu}) = \alpha(p)$. Then, $R_{p,\vec{\mu}}(N, m_0)$ is identifier sound iff (N, m_0) is identifier sound.

Proof:
Let $(N', m_0') = R_{p,\vec{\mu}}(N, m_0)$. Define $Q \subseteq \mathbb{M}(N) \times \mathbb{M}(N')$ such that $(m, m') \in Q$ iff $m(q) = m'(q)$ for all places $q \in P \setminus \{p\}$ and $m'(p_i) + m'(p_f) = m(p)$, i.e., Q is the bisimulation relation as defined in the previous lemma. Then, the statement is a direct consequence of $Id(m) = Id(m')$ for all $(m, m') \in Q$ and the bisimulation. □

Lemma 5.7. Let (N, m_0) be a marked t-PNID with $N = (P, T, F, \alpha, \beta)$, $p \in P$ be a place to expand and $\vec{\mu} \in \mathcal{V}^*$ be a variable vector. Then (N, m_0) is identifier sound iff $R_{p,\vec{\mu}}(N, m_0)$ is identifier sound.

Proof:
Let $(N', m_0') = R_{p,\vec{\mu}}(N, m_0)$, and let $Q \subseteq \mathbb{M}(N) \times \mathbb{M}(N')$ be the bisimulation relation of Lm. 5.5. Let t be the transition added by the place extension rule. Then $t \notin C_N(\lambda)$ for any type $\lambda \in type_\Lambda(N)$. As $Id(m_1) = Id(m_2)$ for all $(m_1, m_2) \in Q$, the statement directly follows from Thm. 4.14. □

5.2.2. Transition Expansion

The second rule is transition expansion, which corresponds to Murata's *fusion of series transitions*. As shown in Fig. 8b, transition t is divided into two transitions, t_e that consumes the tokens, and a second transition t_c that produces the tokens. The two transitions are connected with a single, fresh place p. Place p should ensure that all variables consumed by the original transition t, are passed to transition t_c, to ensure that t_c can produce the same tokens as transition t in the original net. In other words, the type of each input place of t is included in the type of the newly added place p. Moreover, transition t_e is also allowed to emit new, fresh identifiers that, however, will be eventually consumed by t_c.

Definition 5.8. (Transition expansion)
Let (N, m) be a marked t-PNID with $N = (P, T, F, \alpha, \beta)$, let $t \in T$, and let $\vec{\lambda} \in \Lambda^*$ and $\vec{\mu} \in (\mathcal{V} \setminus Emit(t))^*$ such that $type_{\mathcal{V}}(x) \in \vec{\lambda}$ and $x \in \vec{\mu}$, for all $x \in In(t)$, and $type_{\mathcal{V}}(\vec{\mu}) = \vec{\lambda}$. The *transition expanded t-PNID* is defined by $R_{t,\vec{\lambda},\vec{\mu}}(N, m) = ((P', T', F', \alpha', \beta'), m)$, where:

- $P' = P \cup \{p\}$ with $p \notin P$; and $T' = (T \setminus \{t\}) \cup \{t_e, t_c\}$ with $t_e, t_c \notin T$;
- $F' = (F \setminus (({}^\bullet t \times \{t\}) \cup (\{t\} \times t^\bullet))) \cup ({}^\bullet t \times \{t_e\}) \cup \{(t_e, p), (p, t_c)\} \cup (\{t_c\} \times t^\bullet)$;
- $\alpha'(p) = \vec{\lambda}$ and $\alpha'(q) = \alpha(q)$ for all $q \in P$;
- $\beta'(f) = [\,\vec{\mu}\,]$ if $f \in \{(t_e, p), (p, t_c)\}$, $\beta'((q, t_e)) = \beta((q, t))$ for $q \in {}^\bullet t$, $\beta'((t_c, q)) = \beta((t, q))$ for $q \in t^\bullet$, and $\beta'(f) = \beta(f)$ otherwise.

Transition t_e is allowed to introduce new variables, but key is that inscription $\vec{\mu}$ contains all input variables of transition t. Consequently, $\vec{\mu}$ encodes the binding of transition t. We use this to prove weak bisimulation between a t-PNID and it transition expanded net. The idea behind the simulation relation Q is that the firing of t_e is postponed until t_c fires. In other words, Q encodes that tokens remain in place q until transition t_c fires.

Lemma 5.9. Given marked t-PNID (N, m_0) with $N = (P, T, F, \alpha, \beta)$, transition $t \in T$, $\vec{\lambda} \in \Lambda^*$ and $\mu \in \mathcal{V}^*$. Let t_e, t_c be the transitions added by the expansion.
Then $\Gamma_{N,m_0} \approx^r \rho_r(\Gamma_{R_{t,\vec{\lambda},\mu}(N,m_0)})$ with $r = \{(t_e, \tau), (t_c, t)\}$.

Proof:
Let $(N', m_0') = R_{t,\lambda,\mu}(N, m_0)$. Then $m_0' = m_0$. Define relation $Q \subseteq \mathbb{M}(N) \times \mathbb{M}(N')$ such that $(m, m') \in Q$ iff $m(q) = m'(q)$ for all places $q \in P \backslash {}^\bullet t$ and $m(q) = m'(q) + \sum_{b \in supp(M'(p))} M'(p)(b) \cdot \rho_{\mu(b)}\beta((q,t))$, where $\mu(b)$ is a shorthand for the binding $\psi : \mathcal{V} \to \mathcal{I}$ with $\psi(x) = b(i)$ iff $\mu(i) = x$ for all $1 \leq i \leq |\mu|$. Then $(m_0, m_0') \in Q$.

($\Rightarrow$) Follows directly from the firing rule, and the construction of μ.

($\Leftarrow$) Let $(m, m') \in Q$ and $(N', m')[u, \psi\rangle(N', \bar{m}')$. We need to show a marking $\bar{m}$ exists such that $m \xrightarrow{(t,\psi)} \bar{m}$ and $(\bar{m}, \bar{m}') \in Q$. If $t_e \neq u \neq t_c$, the statement holds by definition of the firing rule. Suppose $u = t_e$, i.e., $r(u) = \tau$. Hence, we need to show that $(m, \bar{m}') \in Q$. Let $q \in {}^\bullet t$. Since $(m, m') \in Q$, we have $m(q) = m'(q) + \sum_{b \in supp(m'(p))} m'(p)(b) \cdot \rho_{\mu(b)}\beta((q,t))$. By the firing rule, we have $\bar{m}'(p) = m'(p) + [\rho_\psi(\mu)]$ and $m'(q) = \bar{m}'(q) + \rho_\psi(\beta((q,t)))$. By construction, ρ_ψ and $\rho_{\mu([\rho_\psi(\mu)])}$ are identical functions. Rewriting gives $m(q) = \bar{m}'(q) + \sum_{b \in supp(\bar{m}'(p))} m'(p)(b) \cdot \rho_{\mu(b)}\beta((q,t))$, and thus $(m, \bar{m}') \in Q$.

Suppose $u = t_c$, i.e., $r(u) = t$ and $[\rho_\psi(\mu)] \leq m'(p)$. Let $q \in {}^\bullet t$. Then $m(q) = m'(q) + \sum_{b \in supp(m'(p))} m'(p)(b) \cdot \rho_{\mu(b)}\beta((q,t))$. Since $\bar{m}'(p) + [\rho_\psi(\mu)] = m'(p)$ and $\rho_\psi(\beta((q,u))) = \rho_{\mu([\rho_\psi(\mu)])}(\beta((q,u)))$, we obtain $m(q) = m'(q) + \left(\sum_{b \in supp(\bar{m}'(p))} \bar{m}'(p)(b) \cdot \rho_{\mu(b)}\beta((q,t))\right) + \rho_\psi\beta((q,t))$. Hence, a marking $\bar{m}$ exists such that $(N, m)[t, \psi\rangle(N, \bar{m})$ and $(\bar{m}, \bar{m}') \in Q$. □

Lemma 5.10. Given marked t-PNID (N, m_0) with $N = (P, T, F, \alpha, \beta)$, transition $t \in T$, $\vec{\lambda} \in \Lambda^*$ and $\mu \in \mathcal{V}^*$. Then (N, m_0) is identifier sound iff $R_{t,\vec{\lambda},\mu}(N, m_0)$ is identifier sound.

Proof:
Define $(N', m_0') = R_{t,\vec{\lambda},\mu}(N, m_0)$ with $N' = (P', T', F', \alpha', \beta')$, let $t_e, t_c \in T' \setminus T$ be the two added transitions and let $q \in P' \setminus P$ be the added place. Let $Q \subseteq \mathbb{M}(N) \times \mathbb{M}(N')$ be the weak bisimulation relation as defined in Lm. 5.9.

Suppose $\lambda \in \Lambda$. If $\lambda \in type_\Lambda(N)$, then $C_N(\lambda) = C_{N'}(\lambda)$ by definition of the transition expansion. As $I(\lambda) \cap Id(m_1) = I(\lambda) \cap Id(m_2)$ for all $(m_1, m_2) \in Q$, the statement directly follows from Thm. 4.14.

Otherwise, if $\lambda \in type_\Lambda(N') \setminus type_\Lambda(N)$, then, for all places $p \in P'$, having $\lambda \in \alpha(p)$ implies that $p = q$, i.e., q is the only place that contains tokens carrying identifiers of type λ, and $t_c \in C_{N'}(\lambda)$. Suppose that there exist a marking $m' \in \mathcal{R}(N', m_0')$, firing sequence η, vector $\vec{\mathtt{id}} \in \mathcal{I}^*$ and $\mathtt{id} \in I(\lambda)$ such that $(N', m_0')[\eta\rangle(N', m')$, $m(q)(\vec{\mathtt{id}}) > 0$ and $\mathtt{id} \in \vec{\mathtt{id}}$. Thus, $\mathtt{id} \in Id(m')$. Then a binding $\psi : \mathcal{V} \to \mathcal{I}$ exists such that $\mathtt{id} \in \text{RNG}(\psi)$ and $(t_e, \psi) \in \eta$. As t_e is an emitting transition for λ, we have that $|\beta'((t_e, \psi))| = 1$, and thus $m'(q)(\vec{\mathtt{id}}) = 1$. By the firing rule and the construction of N', we have that $m'(q)(\psi(\beta'((q, t_c)))) > 0$, i.e., $(N', m')[t_c, \psi\rangle$. Hence, a marking $m'' \in \mathbb{M}(N')$ exists with $(N', m')[t_c, \psi\rangle(N', m'')$ such that $\mathtt{id} \notin Id(m'')$. Hence, N' is weakly λ-terminating. As $q^\bullet = \{t_c\}$,

transition t_c is the only transition which can remove identifiers of type λ, and thus N' is also proper λ completing. □

5.2.3. Place Duplication

Whereas the previous two rules introduced ways to extend sequences, the third rule introduces parallelism by duplicating a place, as shown in Figure 8c. It is based on the *fusion of parallel transitions* reduction rule of Murata. For t-PNIDs, duplicating a place has an additional advantage: as all information required for passing the identifiers is already guaranteed, the duplicated place can have any place type. Transition t can emit new identifiers, provided that transition u does not already emit these.

Definition 5.11. (Duplicate place)
Let (N, m) be a marked t-PNID with $N = (P, T, F, \alpha, \beta)$, let $p \in P$, such that $m(p) = \emptyset$, and some transitions $t, u \in T$ exist with ${}^\bullet p = \{t\}$, $t^\bullet = \{p\}$, $p^\bullet = \{u\}$ and ${}^\bullet u = \{p\}$. Let $\vec{\lambda} \in \Lambda^*$ and $\vec{\mu} \in (\mathcal{V} \setminus \mathit{Emit}(u))^*$ such that $\mathit{type}_{\mathcal{V}}(\mu) = \lambda$. Its *duplicated place t-PNID* is defined by $D_{p,\lambda,\mu}(N, m) = ((P', T, F', \alpha', \beta'), m)$, where:

- $P' = P \cup \{q\}$, with $q \notin P$, and $F' = F \cup \{(t, q), (q, u)\}$;
- $\alpha' = \alpha \cup \{q \mapsto \vec{\lambda}\}$ and $\beta' = \beta \cup \{(t, q) \mapsto [\mu], (q, u) \mapsto [\mu]\}$.

As the duplicated place cannot hamper the firing of any transition, all behavior is preserved by a strong bisimulation on the identity mapping.

Lemma 5.12. Given a marked t-PNID (N, m_0) with $N = (P, T, F, \alpha, \beta)$, place $p \in P$, $\vec{\lambda} \in \Lambda^*$ and $\mu \in \mathcal{V}^*$. Then $\Gamma_{N,m_0} \sim^r \Gamma_{D_{p,\vec{\lambda},\mu}(N,m_0)}$.

Proof:
Let $(N', m_0') = D_{p,\vec{\lambda},\mu}(N, m_0)$. Define relation $Q \subseteq \mathbb{M}(N) \times \mathbb{M}(N')$ such that $(m, m') \in Q$ iff $m(q) = m'(q)$ for all places $q \in P$. The bisimulation relation trivially follows from the firing rule. □

Lemma 5.13. Let (N, m_0) be a marked t-PNID with $N = (P, T, F, \alpha, \beta)$, place $p \in P$, $\lambda \in \Lambda^*$ and $\mu \in \mathcal{V}^*$. Then (N, m_0) is identifier sound iff $D_{p,\vec{\lambda},\mu}(N, m_0)$ is identifier sound.

Proof:
Let $(N', m_0') = D_{p,\vec{\lambda},\mu}(N, m_0)$ with $N' = (P', T', F', \alpha', \beta')$, let $q \in P' \setminus P$ be the place added by the place duplication rule, and let $Q \subseteq \mathbb{M}(N) \times \mathbb{M}(N')$ be the bisimulation relation of Lm. 5.15.

($\Rightarrow$) Let $\lambda \in \mathit{type}_\Lambda(N)$ be a type. Then, $\lambda \in \mathit{type}_\Lambda(N')$ and $C_N(\lambda) = C_{N'}(\lambda)$ by definition of the place duplication rule. As $I(\lambda) \cap \mathit{Id}(m_1) = I(\lambda) \cap \mathit{Id}(m_2)$ for all $(m_1, m_2) \in Q$, the statement directly follows from Thm. 4.14.

($\Leftarrow$) Let $\lambda \in \mathit{type}_\Lambda(N')$ be a type. If $\lambda \in \mathit{type}_\Lambda(N)$, then the statement directly follows from Thm. 4.14 as $C_N(\lambda) = C_{N'}(\lambda)$ and $I(\lambda) \cap \mathit{Id}(m_1) = I(\lambda) \cap \mathit{Id}(m_2)$ for all $(m_1, m_2) \in Q$. Otherwise $\lambda \notin \mathit{type}_\Lambda(N)$. Then, by definition of the place duplication, it must be that $\lambda \in \alpha(q)$.

Then $E_{N'}(\lambda) = u$ and $C_{N'}(\lambda) = u$, where $(t,q),(q,u) \subseteq F'$. Suppose there there exist a marking $m' \in \mathcal{R}(N', m'_0)$, firing sequence η, an identifier vector $\vec{\mathtt{id}} \in \mathcal{I}^*$ and identifier $\mathtt{id} \in I(\lambda)$ such that $(N', m'_0)[\eta\rangle(N', m')$, $\mathtt{id} \in \vec{\mathtt{id}}$ and $m'(q)(\vec{\mathtt{id}}) > 0$. Then a binding $\psi : \mathcal{V} \to \mathcal{I}$ exists such that $\mathtt{id} \in \textsc{rng}(\psi)$ and $(t,\psi) \in \eta$. As t is an emitting transition for λ, then $|\beta'((t,\psi))| = 1$, i.e., $m'(q)(\vec{\mathtt{id}}) = 1$. By the firing rule and the construction of N', it holds that $m'(q)(\psi(\beta'((q,u)))) > 0$, and thus $(N', m')[u, \psi\rangle$. Hence, there exists a marking $m'' \in \mathbb{M}(N')$ such that $(N', m')[u, \psi\rangle(N', m'')$. Then $\mathtt{id} \notin Id(m'')$. Hence, N' is weakly λ-terminating. As $q^\bullet = \{u\}$, transition u is the only transition which can remove identifiers of type λ, and hence N' is also proper λ-completing. □

5.2.4. Transition Duplication

As already recognized by Berthelot [31], if two transitions have an identical preset and postset, one of these transitions can be removed while preserving liveness and boundedness. Murata's fusion of parallel places is a special case of this rule, requiring that the preset and postset are singletons. For t-JNs, this results in the duplicate transition rule: any transition may be duplicated, as shown in Figure 8d. As duplication should not hamper the behavior of the original net, we require that the inscriptions of the duplicated transition are identical to the original transition.

Definition 5.14. (Duplicate transition)
Let (N, m) be a marked t-PNID with $N = (P, T, F, \alpha, \beta)$, and let $t \in T$ such that some places $p, q \in P$ exist with ${}^\bullet t = \{p\}$ and $t^\bullet = \{q\}$. Its *duplicated transition t-PNID* is defined by $D_t(N, m) = ((P, T', F', \alpha, \beta'), M)$, where:

- $T' = T \cup \{u\}$, with $u \notin T$, and $F' = F \cup \{(p,u),(u,q)\}$;
- $\beta'((p,u)) = \beta((p,t))$, $\beta((u,q)) = \beta((t,q))$ and $\beta'(f) = \beta(f)$ for all $f \in F$.

As the above rule only duplicates $t \in T$, the identity relation on markings is a strong rooted bisimulation. The proof is straightforward from the definition.

Lemma 5.15. Given a marked t-PNID (N, m_0) with $N = (P, T, F, \alpha, \beta)$, and transition $t \in T$. Then $\Gamma_{N,m_0} \sim^r \rho_{\{(u,t)\}}(\Gamma_{D_t(N,m_0)})$.

Proof:
Let $(N', m'_0) = D_t(N, m_0)$. Define relation $Q \subseteq \mathbb{M}(N) \times \mathbb{M}(N')$ such that $(m, m') \in Q$ iff $m(p) = m'(p)$ for all places $p \in P$. The bisimulation relation trivially follows from the firing rule. □

Lemma 5.16. Let (N, m_0) be a marked t-PNID with $N = (P, T, F, \alpha, \beta)$ and transition $t \in T$. Then (N, m_0) is identifier sound iff $D_t(N, m_0)$ is identifier sound.

Proof:
Let $(N', m'_0) = D_t(N, m_0)$, and let $Q \subseteq \mathbb{M}(N) \times \mathbb{M}(N')$ be the bisimulation relation of Lm. 5.15. Then $Id(m_1) = Id(m_2)$ for all $(m_1, m_2) \in Q$. If $t \notin C_N(\lambda)$ then the statement directly follows from Thm. 4.14. Otherwise, i.e., $t \in C_N(\lambda)$, then $C_N(\lambda) = C_N(\lambda) \cup \{u\}$. By Lm 4.13, N is weakly λ terminating. As $\beta_{N'}((t,p)) = \beta_{N'}((u,p))$ and $\beta_{N'}((p,t)) = \beta_{N'}((p,u))$ for all places $p \in P_{N'}$, proper type completion cannot distinct firing transition t from transition u. Hence, proper λ completion follows from the proof of Lm 4.12, and thus, N is identifier sound. □

5.2.5. Adding Identity Transitions

In [31], Berthelot classified a transition t with an identical preset and postset, i.e., ${}^\bullet t = t^\bullet$ as irrelevant, as its firing does not change the marking. The reduction rule *elimination of self-loop transitions* is a special case, as Murata required these sets to be singletons. We now introduce the fifth rule allowing the addition of a self-loop transition, as depicted in Figure 8e.

Definition 5.17. (Self-loop addition)
Let (N, m) be a marked t-PNID with $N = (P, T, F, \alpha, \beta)$, and let $p \in P$. Its *self-loop added t-PNID* is defined by $A_p(N, m) = ((P, T', F', \alpha, \beta'), m)$, where:

- $T' = T \cup \{t\}$, with $t \notin T$, and $F' = F \cup \{(p, t), (t, p)\}$;
- $\beta'((p, t)) = \beta'((t, p)) = [\,\vec{\mu}\,]$ with $\vec{\mu} \in \mathcal{V}^*$ such that $type_{\mathcal{V}}(\mu) = \alpha(p)$, and $\beta'(f) = \beta(f)$ otherwise.

Similar to the duplicate transition rule, the self-loop addition rule does not introduce new behavior, except for silent self-loops. Hence, the identity relation on markings is a weak rooted bisimulation.

Lemma 5.18. Given a marked t-PNID (N, m_0) with $N = (P, T, F, \alpha, \beta)$, and place $p \in P$. Then $\Gamma_{N,m_0} \approx^r \hat{\mathfrak{H}}_{\{t\}}(\Gamma_{A_p(N,m_0)})$ with t the added self-loop transition.

Proof:
Let $(N', m_0') = A_p(N, m_0)$. Define relation $Q \subseteq \mathbb{M}(N) \times \mathbb{M}(N')$ such that $(m, m') \in Q$ iff $m(p) = m'(p)$ for all places $p \in P$. The bisimulation relation trivially follows from the firing rule. □

Lemma 5.19. Let (N, m_0) be a marked t-PNID with $N = (P, T, F, \alpha, \beta)$ and place $p \in P$. Then (N, m_0) is identifier sound iff $A_p(N, m_0)$ is identifier sound.

Proof:
Let $(N', m_0') = A_p(N, m_0)$ and let $Q \subseteq \mathbb{M}(N) \times \mathbb{M}(N')$ be the bisimulation relation of Lm. 5.15. Note that $C_N(\lambda) = C_{N'}(\lambda)$ for all $\lambda \in type_\Lambda(N)$, since the added self-loop transition does not remove any identifier. As $Id(m_1) = Id(m_2)$ for all $(m_1, m_2) \in Q$, the statement directly follows from Thm. 4.14. □

5.2.6. Identifier Introduction

The first five rules preserve the criteria of block-structured WF-nets. Murata's *elimination of self-loop places* states that adding or removing a marked place with identical preset and postset does preserve liveness and boundedness. This rule is often used to introduce a fixed resource to a net, i.e., the number of resources is determined in the initial marking. Instead, identifier introduction adds dynamic resources, as shown in Figure 8f: transition t_e emits new identifiers as its inscription uses only "new" variables (i.e., those that have not been used in the net), and place p works like a storage of the available resources, which can be removed by firing transition t_c.

Definition 5.20. (Identifier Introduction)
Let (N, m) be a marked t-PNID with $N = (P, T, F, \alpha, \beta)$, let $t \in T$, let $\vec{\lambda} \in (\Lambda \setminus type_P(N))^*$ and $\vec{\mu} \in \mathcal{V}^*$ such that $type_{\mathcal{V}}(\vec{\mu}) = \vec{\lambda}$. The *Identifier introducing t-PNID* is defined by $A_{t,\vec{\lambda},\vec{\mu}}(N, m) = ((P', T', F', \alpha', \beta'), m)$, where:

- $P' = P' \cup \{p\}$ and $T' = T \cup \{t_e, t_c\}$, for $p \notin P$ and $t_e, t_c \notin T$, and $F' = F \cup \{(p, t), (t, p), (t_e, p), (p, t_c)\}$;
- $\alpha' = \alpha \cup \{p \mapsto \vec{\lambda}\}$ and $\beta' = \beta \cup \{(p, t) \mapsto [\vec{\mu}], (t, p) \mapsto [\vec{\mu}], (t_e, p) \mapsto [\vec{\mu}], (p, t_c) \mapsto [\vec{\mu}]\}$;

Lemma 5.21. Given a marked t-PNID (N, m_0) with $N = (P, T, F, \alpha, \beta)$, transition $t \in T$, $\vec{\lambda} \in \Lambda^*$ and $\vec{\mu} \in (\mathcal{V} \setminus Var(t))^*$. Then $\Gamma_{N,m_0} \approx^r \hat{\mathfrak{H}}_{\{t_e,t_c\}}(\Gamma_{A_{t,\vec{\lambda},\vec{\mu}}(N,m_0)})$ with t_e, t_c being the added transitions.

Proof:
Let $N' = (P', T', F', \alpha', \beta')$. Define $Q \subseteq \mathbb{M}(N) \times \mathbb{M}(N')$ such that $(m, m') \in Q$ iff $m(q) = m'(q)$ for all $q \in P$.

($\Rightarrow$) Suppose $(N, m)[u, \psi\rangle(N, \bar{m}')$ and $(m, m') \in Q$. If $u \neq t$, the statement directly follows from the firing rule. Same holds for the case when $u = t$ and t is enabled in m'. If $u = t$ and t is not enabled in m', then a marking m'' and binding ψ' exist such that $(N', m')[t_e, \psi'\rangle(N, m'')$. Then $m''(p) > \emptyset$, $(m, m'') \in Q$, and $(N', m'')[u, \psi\rangle$. Hence, markings $\bar{m}''$ and $\bar{m}'$ exist such that $(N', m'')[t, \psi\rangle(N', \bar{m}'')[t_c, \psi'\rangle(N, \bar{m}')$, and $(m, \bar{m}''), (m', \bar{m}') \in Q$.

($\Leftarrow$) Follows directly from the firing rule. □

As shown in [12], unbounded places are width-bounded, i.e., they can carry only boundedly many distinct identifiers, or depth-bounded, i.e., for each identifier, the number of tokens carrying that identifier is bounded, or both. The place added by the identifier creation rule is by definition width-unbounded, as it has an empty preset. However, it is identifier sound, and thus depth-bounded, as shown in the next lemma.

Lemma 5.22. Given a marked t-PNID (N, m) with $N = (P, T, F, \alpha, \beta)$. Then $A_{t,\vec{\lambda},\vec{\mu}}(N, m)$ is identifier sound iff (N, m) is identifier sound.

Proof:
Let $(N', m') = A_{t,\vec{\lambda},\vec{\mu}}(N, m)$, let $p \in P' \setminus P$, and let $\lambda \in type_\Lambda(N')$.

($\Rightarrow$) Suppose (N, m) is identifier sound. Let $\bar{m} \in \mathbb{M}(N')$ and $\eta \in (T' \times (\mathcal{V} \to \mathcal{I}))^*$ such that $(N', m')[\eta\rangle(N', \bar{m})$. Let $\mathtt{id} \in Id(\bar{m}) \cap I(\lambda)$. If $\lambda \in type_\Lambda(N)$, weak λ-termination and proper λ-completion follow from Lm. 5.21. Suppose $\lambda \notin type_\Lambda(N)$, i.e., $\lambda \in \vec{\lambda}$. By construction of N', we have $\lambda \in \alpha(q)$ implies $p = q$ for all places $q \in P'$, $E_{N'}(\lambda) = \{t_e\}$ and $C_{N'}(\lambda) = \{t_c\}$. By the firing rule, we have $\mathtt{id} \in \vec{a}$ and $\mathtt{id} \in \vec{b}$ imply $\vec{a} = \vec{b}$ for all $\vec{a}, \vec{b} \in supp(m(p))$. Again by the firing rule, $m(p)(\vec{a}) \leq 1$ for all $\vec{a} \in supp(m(p))$. In other words, there is only one token carrying identifier $\mathtt{id}$. Let $\vec{\mathtt{id}} \in \mathtt{C}(p)$ such that $\mathtt{id} \in \vec{\mathtt{id}}$ and $m(p)(\vec{\mathtt{id}}) > 0$. Then $m(p)(\vec{\mathtt{id}}) = 1$. Thus, a binding ψ exists such that $(t_e, \psi) \in \eta$ and $\rho_{\vec{\mu}}(\psi) = \vec{\mathtt{id}}$. By construction of N', a marking $\bar{m}'$ exists such that $(N', \bar{m})[t_c, \psi\rangle(N', \bar{m}')$. Then $\mathtt{id} \notin Id(\bar{m}')$. Hence, (N', m') is weakly λ-terminating. It is proper λ-completing since there is only one token carrying identifier $\mathtt{id}$.

($\Leftarrow$) Suppose (N', m') is identifier sound. If $\lambda \in type_\Lambda(N)$, weak λ-termination and proper λ-completion follow from Thm. 4.14. In case $\lambda \notin type_\Lambda(N)$, it is weakly λ-terminating, since $E_N(\lambda) = \emptyset$, and properly λ-completing since $C_N(\lambda) = \emptyset$. □

5.2.7. Soundness for Typed Jackson Nets

Any net that can be reduced to a net with a single transition using these rules is called a typed Jackson Net (t-JN).

Definition 5.23. The class of *typed Jackson Nets* $\mathcal{T}$ is inductively defined by:

- $((\emptyset, \{t\}, \emptyset, \emptyset, \emptyset), \emptyset) \in \mathcal{T}$;
- if $(N, M) \in \mathcal{T}$, then $R_{p,\vec{\mu}}(N, M) \in \mathcal{T}$;
- if $(N, M) \in \mathcal{T}$, then $R_{t,\vec{\lambda},\vec{\mu}}(N, M) \in \mathcal{T}$;
- if $(N, M) \in \mathcal{T}$, then $D_{p,\vec{\lambda},\vec{\mu}}(N, M_0) \in \mathcal{T}$;
- if $(N, M) \in \mathcal{T}$, then $D_t(N, M) \in \mathcal{T}$;
- if $(N, M) \in \mathcal{T}$, then $A_p(N, M) \in \mathcal{T}$;
- if $(N, M) \in \mathcal{T}$, then $A_{t,\vec{\lambda},\vec{\mu}}(N, M) \in \mathcal{T}$.

As any t-JN reduces to a single transition, and each construction rule goes hand in hand with a bisimulation relation, any liveness property is preserved. Consequently, any t-JN is identifier sound and live.

Theorem 5.24. Any typed Jackson Net is identifier sound and live.

Proof:
We prove the statement by induction on the structure of t-JNs. The statement holds trivially for the initial net, $((\emptyset, \{t\}, \emptyset, \emptyset, \emptyset), \emptyset)$. Suppose $(N', M') \in \mathcal{T}$ is identifier sound. We show that applying any of the construction rules on (N', M') preserves identifier soundness:

- Suppose $(N, M) = R_{p,\vec{\mu}}(N', M')$. The statement follows directly from Lm. 5.7.
- Suppose $(N, M) = R_{t,\vec{\lambda},\vec{\mu}}(N', M')$. The statement follows directly from Lm. 5.10.
- Suppose $(N, M) = D_{p,\vec{\lambda},\vec{\mu}}(N', M')$. The statement follows directly from Lm. 5.13.
- Suppose $(N, M) = D_t(N', M')$. The statement follows directly from Lm. 5.16.
- Suppose $(N, M) = A_p(N', M')$. The statement follows directly from Lm. 5.19.
- Suppose $(N, M) = A_{t,\vec{\lambda},\vec{\mu}}(N', M')$. The statement follows directly from Lm. 5.22.

□

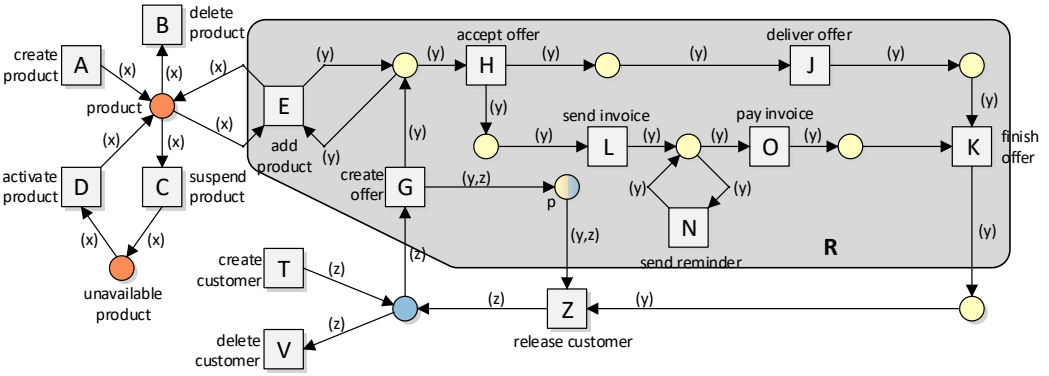
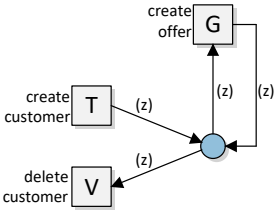
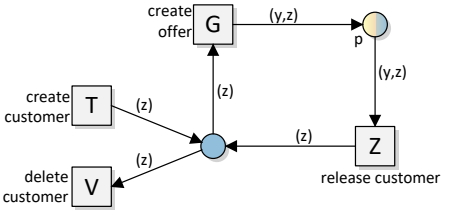


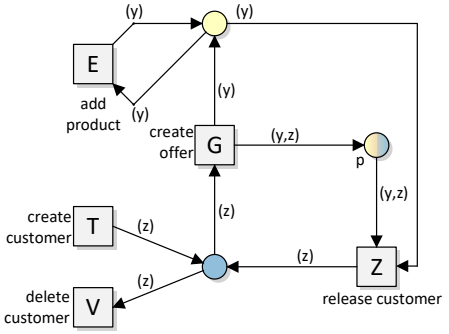
Figure 9: The example of the retailer shop as a typed Jackson Net.



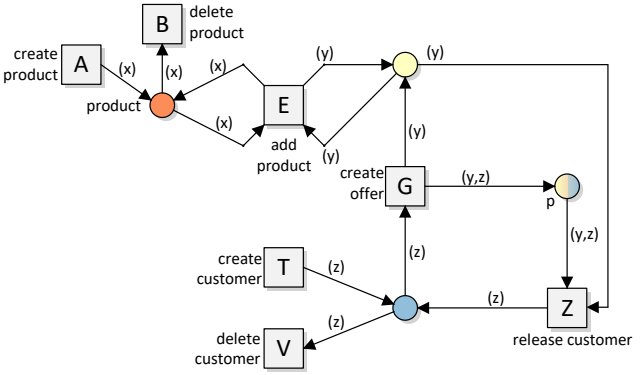
(a) Step 2



(b) Step 3



(c) Step 5



(d) Step 6

Figure 10: Several intermediate steps while creating the typed Jackson Net of Fig. 9.

To solve the problem of the running example, several solutions exist. One solution is shown in Figure 9, which is a t-JN. Several intermediate construction steps are shown in Fig. 10. The modeler starts with transition T, "create customer". The net has no identifiers yet. Next, transition T is expanded using place type $\langle customer \rangle$, i.e., a place and transition V are added. A self loop is added to the newly created place (transition G), which results in the net depicted in Fig. 9a. The next step introduces place p, and is shown in Fig. 9b: transition G is expanded using place type $\langle order, customer \rangle$.

Duplicating place p allows to create a place with place type $\langle customer \rangle$. The net depicted in Fig. 9c shows the net after adding another self-loop (transition E). The identifier introduction rule allows the modeler to add a product life cycle, so that transition E can add actual products, resulting in the net depicted in Fig. 9d. Now, all required identifiers are present, and transitions H, K, J, L, O and N are added using the place type $\langle customer \rangle$, which results in the net depicted in Fig. 9. As only the types Jackson rules are used, the net is guaranteed to be identifier sound and live.

5.3. Workflow refinement

A well-known refinement rule is workflow refinement [10]. In a WF-net, any place may be refined with a generalized sound WF-net. If the original net is sound, then the refined net is sound as well. In this section, we present a similar refinement rule. Given a t-PNID, any place may be refined by a generalized sound WF-net. In the refinement, each place is labeled with the place type of the refined place, and all arcs in the WF-net are inscribed with the same variable vector.

Definition 5.25. (Workflow refinement)
Let $L = (P_L, T_L, F_L, \alpha_L, \beta_L)$, be a t-PNID, $p \in P_L$ a place, and $N = (P_N, T_N, F_N, W_N, in, out)$ be a WF-net. *Workflow refinement* is defined by $L \oplus_p N = (P, T, F, \alpha, \beta)$, where:

- $P = (P_L \setminus \{p\}) \cup P_N$ and $T = T_L \cup T_N$;
- $F = (F_L \cap ((P \times T) \cup (T \times P))) \cup F_N \cup \{(t, in) \mid t \in {}^\bullet p\} \cup \{(out, t) \mid t \in p^\bullet\}$;
- $\alpha(q) = \alpha_L(q)$ for $q \in P_L \setminus \{p\}$, and $\alpha(q) = \alpha_L(p)$ for $q \in P_N$;
- $\beta(f) = \beta_L(f)$ for $f \in F_L$, $\beta(f) = [\vec{\mu}]^{(W(f))}$ for $f \in F_N$ and $type_{\mathcal{V}}(\vec{\mu}) = \alpha(p)$, $\beta((t, in)) = \beta((t, p))$ for $t \in {}^\bullet p$ and $\beta((out, t)) = \beta((p, t))$ for $t \in p^\bullet$.

Generalized soundness of a WF-Net ensures that any number of tokens in the initial place are "transferred" to the final place. As shown in Section 5.1, the EC-closure of a sound WF-net is identifier sound and live. A similar approach is taken to show that the refinement is weakly bisimilar to the original net. Analogously to [10], the bisimulation relation is the identity relation, except for place p. The relation maps all possible token configurations of place p to any reachable marking in the WF-net, given p's token configuration.

Lemma 5.26. Let $L = (P_L, T_L, F_L, \alpha_L, \beta_L)$ be a t-PNID with initial marking m_0, let $p \in P_L$ be a place s.t. $m_0(p) = \emptyset$, and let $N = (P_N, T_N, F_N, W_N, in_N, out_N)$ be a WF-net. If N is generalized sound, then $\Gamma_{L,m_0} \approx^r \hat{\mathfrak{H}}_{T_N}(\Gamma_{L \oplus_p N, m_0})$.

Proof:
For simplicity, we start by defining a type extension of N as a t-PNID $N' = (P_{N'}, T_{N'}, F_{N'}, \alpha, \beta)$, where $type(\vec{v}) = \vec{\lambda}$, $\alpha(p) = \vec{\lambda}$ for all places $p \in P_N$, and $\beta(f) = \vec{v}^{W(f)}$ for all $f \in F_N$, and $\beta((t_e, in)) = \beta((out, t_c)) = [\vec{v}]$.

To prove bisimilarity, we define $R = \{(M, M' + m) \mid M \in \mathcal{R}(L, M_0), M' \in \mathcal{A}(M), m \in \mathcal{B}(M)\}$ where

- $\mathcal{A}(M) := \{M' \mid M' \in \mathcal{R}(L, M_0), M'(p) = \emptyset$ and $\forall q \in P_L \setminus \{p\} : M'(q) = M(q)\}$, and

- $\mathcal{B}(M) := \{m \mid m \in \mathcal{R}(N', m_0''), m_0''(in) = M(p) \text{ and } \forall q \in P_{N'} \setminus \{p\} : m_0''(q) = \emptyset\}$.

Intuitively, $\mathcal{B}(M)$ is essentially the $\vec{\lambda}$-typed set of reachable markings of N for a fixed k-tokens in in, where such tokens in N' are provided by M (more specifically, by $M(p)$).

($\Rightarrow$) Let $(M, M' + m) \in R$ and $M[t, \psi\rangle\bar{M}$. We need to show that there exists $\bar{M}'$ and $\bar{m}$ s.t. $(M' + m) \xrightarrow{(t,\psi)} (\bar{M}' + \bar{m})$ and $(\bar{M}, \bar{M}' + \bar{m}) \in R$. To this end, we consider the following cases.

(i) If $t \notin {}^\bullet p$ (or $p \notin {}^\bullet t$), then $M'(q) = M(q)$ for all $q \in P_L \setminus p$ (follows from the definition of $\mathcal{A}(M)$), and thus t is also enabled in $M'(q)$ and $M(q) \geq \beta((q, t))$. Then by the firing rule there exists $\bar{M}'$ s.t. $(M' + m)[t, \psi\rangle(\bar{M}' + m)$ and $\bar{M}'(q) = \bar{M}(q)$ for all $q \in P_L$. Thus, $(M', \bar{M}' + m) \in R$.

(ii) If $t \in {}^\bullet p$, then, since $M'(q) = M(q)$ for all $q \in P_L \setminus p$, t must be enabled in M'. by the refinement construction from Definition 5.25, t is enabled regardless the marking of in. By the firing rule, there exists $\bar{M}'$ and $\bar{m}$ such that $(M' + m)[t, \psi\rangle(\bar{M}' + \bar{m})$, $\bar{M}'(q) = \bar{M}(q)$ for all $q \in P_L \setminus p$, and $\bar{m}(in) = \bar{M}(p) + m(in)$. Moreover, by the definition of R, in can be marked with arbitrarily many tokens from $M(p)$. Thus, $(M', \bar{M}' + \bar{m}) \in R$.

(iii) If $p \in {}^\bullet t$ and $\rho_\psi(\beta((p, t))) = \vec{\mathtt{id}}$, then, given that N is generalized sound and by applying Lemma 5.2, there exists a firing sequence η for N' that carries identifier $\vec{\mathtt{id}}$ to out. This means that, by construction, $M'(q) = M(q)$, for all $q \in P_L$, and $m(out)(\vec{\mathtt{id}}) = M(p)(\vec{\mathtt{id}})$. Hence, t is enabled in $(M' + m)$ under binding ψ' that differs from ψ everywhere but on place out. By the firing rule, there exists $(\bar{M}' + \bar{m})$ s.t. $(M' + m)[t, \psi'\rangle(\bar{M}' + \bar{m})$ and $(M, \bar{M}' + \bar{m}) \in R$.

(iv) If $t \in {}^\bullet p \cap p^\bullet$, then $M(p) \neq \emptyset$ (since $M[t, \psi\rangle\bar{M}$). Assume that $\rho_\psi(\beta((t, p))) = \vec{\mathtt{id}}_1$ and $\rho_\psi(\beta((p, t))) = \vec{\mathtt{id}}_2$. By construction, we know that $M'(q) = M(q)$ for all $q \in P_L$ and m marks some of the places in P_N. Since N is generalized sound and by Lemma 5.2, we can safely assume that $\vec{\mathtt{id}}_2 \in m(p)$ (otherwise, we can apply the reasoning from the previous case). Then it easy to see that, by construction, t is enabled in $(M' + m)$ under the same binding ψ. Thus, by the firing rule there exists $\bar{M}'$ s.t. $(M' + m)[t, \psi\rangle(\bar{M}' + \bar{m})$, where $\bar{M}'(q) = \bar{M}(q)$ for all $q \in P_L$, $\bar{m}(in) = m(in) + [\vec{\mathtt{id}}_1{}^{\beta((t,in))}]$, $\bar{m}(out) = m(out) - [\vec{\mathtt{id}}_2{}^{\beta((out,t))}]$ and $m(w) = \bar{m}(w)$ for all $w \in P_N$. It is easy to see that $(M', \bar{M}' + m) \in R$.

($\Leftarrow$) Let $(M' + m)[t, \psi\rangle(\bar{M}' + \bar{m})$ and $(M, M' + m) \in R$. If $t \in T_L$, then this can be proven by analogy with the previous cases (that is, we need to consider all possible relations of t and p). If $t \in T_N$, then $(M' + m)[t, \psi\rangle(M' + \bar{m})$ and $(\bar{M}, M' + \bar{m}) \in R$, where $\bar{M}(q) = M'(q)$, for all $q \in P_L$, and $\bar{M}(p) = M(p)$. □

As a consequence of the bisimulation relation, the refinement is identifier sound and live if the original net is identifier sound.

Theorem 5.27. Let (L, M) be a marked t-PNID and N be a generalized sound WF net. Then (L, M) is identifier sound and live iff $(L \oplus N, M)$ is identifier sound and live.

The refinement rule allows to combine the approaches discussed in this section. For example, a designer can first design a net using the construction rules of Section 5.2, and then design generalized WF-nets for specific places. In this way, the construction rules and refinement rules ensure that the designer can model systems where data and processes are in resonance.

6. Enriching PNIDs with resources

In the previous section we discussed pattern-based correctness criteria, which allow to construct PNID models that are sound by design. We now consider arbitrary PNIDs, and study how they can be enriched with *resources*, introducing a dedicated property, called *conservative resource management*, which captures that the net suitably employs resources. Following a similar approach, in spirit, to that of Section 5, we define a modelling guideline, called *resource closure*, which takes as input a PNID and indicates how to enrich it with resources through a well-principled approach. We then show that, by construction, if the input PNID is sound, then all its possible resource closures do not only maintain soundness, but they also guarantee that resources are conservatively managed. In addition, we prove that such resource closures are also bounded, and discuss the implications on the analysis of this class of PNIDs.

6.1. Resource-aware PNIDs

As customary for Petri nets, we model resource types as (special) places. However, differently from typical approaches like [32, 33, 20], where resources are represented as indistinguishable (black) tokens populating such places, we assign identifiers to resources. This allows one to explicitly track how resources participate to the execution, and in particular how they relate to the different objects. At the same time, this poses a conceptual question: are different copies of the same identifier in distinct tokens representing different actual resources, or distinct references to the same resource? We opt for the latter approach, as it is the one that fully complies with this *named approach* to resource management. As a consequence of this choice, we blur in the section the distinction between resource and resource identifier, using the two terms interchangeably.

Technically, from now on we assume that Λ is partitioned into two sets: Λ^o for object types, and Λ^r for resource types. Given a resource type $\eta \in \Lambda^r$, we call its identifiers *(η-)resources*. We then simply define a *resource-aware* t-PNID as a t-PNID with some distinguished places, each being of a certain resource type.

Definition 6.1. (Resource-aware t-PNID)
A t-PNID $N = (P, T, F, \alpha, \beta)$ is *resource-aware* if there exists at least one place $p \in P$ such that $\alpha(p) \in \Lambda^r$. We refer to the non-empty subset $P^r = \{p \in P \mid \alpha(p) \in \Lambda^r\}$ of P as the set of *resource places* of N.

The initial marking of a resource-aware t-PNID hence identifies which resources are available per resource type. Consistently with the named approach to resources, every resource should be present at most once in the initial marking.

Places typed by the combination of one or more object types and one resource type are used to establish relations between (tuples of) objects and corresponding resources, which we can interpret as *resource assignments*. For example, given an object type $Order$ and a resource type $Clerk$, a token carrying pair $\langle o, c \rangle$ with $o \in Order$ and $c \in Clerk$ represents that order o is assigned to clerk c.

In an unrestricted t-PNID, resources and resource assignments can be freely manipulated, generating new resources along the execution, assigning the same resource to multiple objects, and establishing arbitrary relations between resources and objects/other resources. To determine whether a t-PNID employs resources properly, we hence introduce a dedicated property that, intuitively, combines two requirements:

- *Resource preservation* - only resources present in the initial marking can be used throughout the execution;
- *Resource exclusive assignment* - in a given marking, each resource can be assigned to at most one object, indicating that the resource is currently responsible for that tuple only.[8]

The first requirement dictates that no resource can be newly generated during the execution; the second one stipulates that at every step, a resource can be responsible for at most one object, possibly carried by multiple tokens.

We formalize these two requirements as follows.

Definition 6.2. (Conservative resource management)
A resource-aware marked t-PNID (N, m_0) with $N = (P, T, F, \alpha, \beta)$ is *managing resources conservatively* if the following two conditions hold.

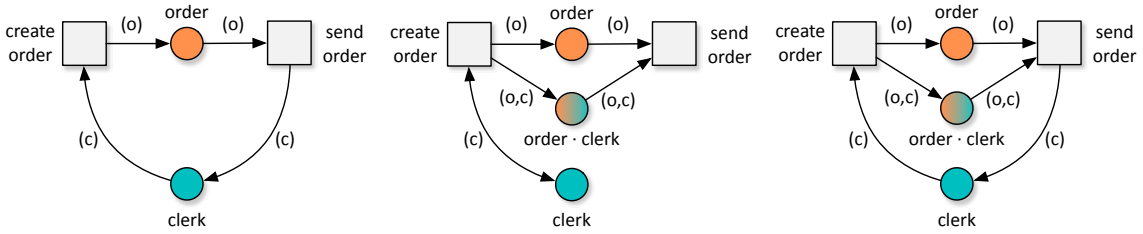
- *Resource preservation*: for every marking $m \in \mathcal{R}(N, m_0)$, resource type $\eta \in \Lambda^r$, and resource $r \in I(\eta) \cap Id(m)$, we have that $r \in Id(m_0)$.
- *Resource exclusive assignment*: for every marking $m \in \mathcal{R}(N, m_0)$, resource type $\eta \in \Lambda^r$, and resource $r \in I(\eta) \cap Id(m)$, there is exactly one tuple $\vec{r} \in supp(m)$ s.t. either $\vec{r} = (r)$ or $\vec{r} = \langle o, r \rangle$ for some object o.

Consider a resource for r present in the initial marking M_0, and a reachable marking M. Two observations are in place regarding Definition 6.2. First, resource exclusive assignment requires that at most one tuple of the form (o, r) exists in the support of M, to express that multiple tokens carrying the same pair (o, r) may indeed exist, while it is not possible to have in the same marking a different tuple of the form (o_2, r) for some $o_2 \neq o$ (which would indicate the simultaneous assignment of r to o and o_2). Second, an active resource r in M can then appear in one and only one of the following forms: either

- (o, r) for some object o – indicating that r is currently assigned to o), or
- (r) - indicating that r is active and not assigned to any object.

Example 6.3. Figure 11 shows three examples of resource-aware t-PNIDs, where place *order* contains objects, and place *clerk* resources.

[8] An analogous treatment of resources can be defined over tuples of objects, instead of single objects.



(a) Resource not preserved (b) Resource not exclusive (c) Conservative resource

Figure 11: Three examples of resource-aware t-PNIDs. Nets (a) and (b) are not managing resources conservatively, as they respective violate the property of resource preservation and that of resource exclusive assignment. Net (c) is instead a positive example that satisfies both properties.

The t-PNID in Figure (11a) attempts to model a setting where every order is managed by a clerk. The main issue here is that there is no information stored in the net about which clerk handles which order. In fact, starting from a marking that indicates which clerks are available, the net violates the property of resource preservation, as when *send order* fires, it brings into the *clerk* place a freshly generated resource (not matching the one previously consumed in *create order* - recall, in fact, that the scope of variables is that of a single transition).

The t-PNID in Figure (11b) explicitly keeps track of the assignments of clerks to orders in a dedicated "synchronization" place. Starting from a marking that indicates which clerks are available, it satisfies resource preservation, as no new clerk identifier is generated, but it violates the property of resource exclusive assignment, since two different order creations may lead to select the same resource twice, assigning it to two different orders.

The t-PNID in Figure (11c) properly handles the assignment of clerks to orders. Starting from a marking that indicates which clerks are available, every time a new order is created, an existing clerk is exclusively assigned to that order. The fact that the same clerk is not reassigned is guaranteed by the fact that the clerk is consumed upon creating the order, and recalled in the assignment place. When the order is sent, its exclusively assigned clerk is released back into the place of (available) clerks, and can be later exclusively assigned to a different order.

By recalling that t-PNIDs evolving tokens that carry pairs of identifiers are Turing-powerful (see [17], and also the proof of Theorem 4.10), and hence every non-trivial property defined over them is undecidable to check, we obtain the following.

Remark 6.4. Verifying whether a marked t-PNID manages resources conservatively is in general undecidable.

To mitigate this negative result, we introduce an approach that drives the modeller in enriching an input t-PNID via resources following a well-principled approach. The approach generalizes the idea introduced in [20] to the more sophisticated case of t-PNIDs, and does so by following the modelling strategy used in Figure 11c. In particular, it aims at capturing the following modelling principles:

1. every object type λ is associated to a dedicated resource type η_λ;

2. each such resource type is used in two places - one just typed with η_λ, to indicate which resources of that type are currently available, the other typed by $\lambda \cdot restype_\lambda$, to keep track of which resources are currently assigned to which objects;
3. every object of type λ gets assigned a resource of type η_λ upon creation, and until the consumption of the object, its resource cannot be assigned to any other object;
4. transitions applied to an object may (or may not) require its resource in isolation from the others (if so, implicitly introducing serialization);
5. upon consumption of an object, its resource may either be permanently consumed as well, or freed and become again available to further assignments.

Technically, we substantiate these intuitive principles through the notion of resource closure.

Definition 6.5. (Resource closure)
Let $N = (P, T, F, \alpha, \beta)$ and $N' = (P', T', F', \alpha', \beta')$ be two t-PNIDs, and $\lambda \in type_\Lambda(N)$ be an object type. We say that N' is a *λ-resource closure* of N if the following conditions hold:

1. $P' = P \cup \{p_r, p_s\}$, where and p_r and p_s are respectively called the *resource* and *assignment* places, and $p_r, p_s \notin P$.
2. $\alpha' = \alpha \cup \{p_r \mapsto \eta \mid \eta \in \Lambda^r \setminus type(N)\} \cup \{p_s \mapsto \lambda \cdot \alpha'(p_r)\}$ extends α by typing p_r with a resource type η not already used in N, and p_s with the combination of the object type λ and the resource type η of p_r.
3. $F' = F \cup F_r^{out} \cup F_r^{in} \cup F_s^{in} \cup F_s^{syn} \cup F_s^{out}$, where:
   (a) $F_r^{out} = \{(p_r, t) \mid t \in E_N(\lambda)\}$, the *output resource flow relation*, indicates that every emitter transition for λ consumes a resource from p_r;
   (b) $F_r^{in} \subseteq \{(t, p_r) \mid t \in C_N(\lambda)\}$, the *input resource flow relation*, indicates that every collector transition for λ *may* return a resource to p_r;
   (c) $F_s^{in} = \{(t, p_s) \mid t \in E_N(\lambda)\}$, the *input assignment flow relation*, indicates that every emitter transition generates an assignment in p_s;
   (d) $F_s^{out} = \{(p_s, t) \mid t \in C_N(\lambda)\}$, the *output assignment flow relation*, indicates that every collector transition consumes an assignment from p_s;
   (e) $F_s^{syn} \subseteq \{(p_s, t), (t, p_s) \mid t \in T \setminus (E_N(\lambda) \cup C_N(\lambda))\}$, the *synchronization assignment flow relation*, indicates that every "internal" (i.e., non-emitting and non-consuming) transition for λ *may* check for the presence of an assignment in p_s.
4. β' is the extension of β satisfying the following conditions:
   (a) $\beta'(a, b) = \beta(a, b)$, if $(a, b) \in F$.
   (b) For every λ-emitter $t \in E_N(\lambda)$, we define $\beta'(p_r, t)$ and $\beta'(t, p_s)$ as described next. Let $X = \{x_1, \ldots, x_n\}$ be the set of distinct variables of type λ mentioned in the inscriptions of outgoing arcs of t, that is, $X = \{x | x \in \beta(t, p) \text{ for some } p \in P\}$. These variables denote the n distinct objects of type λ created upon firing t. We then need to consume n distinct resources of type η and establish the corresponding assignments: $\beta(p_r, t) = (r_1) + \ldots + (r_n)$ and $\beta(t, p_s) = (x_1, r_1) + \ldots + (x_n, r_n)$, where $r_1, \ldots, r_n$ are n distinct (resource) variables of type η.

(c) A symmetric approach is used to define $\beta'(p_s, t)$ and (in case $(t, p_r) \in F_r^{in}$ is defined) $\beta'(t, p_r)$ for every λ-collector $t \in C_N(\lambda)$ (based on the incoming arcs of t).

(d) For every (internal) transition $t \in T \setminus (E_N(\lambda) \cup C_N(\lambda))$ such that $\{(p_s, t), (t, p_s)\} \subseteq F_s^{syn}$, we define $\beta'(p_s, t)$ and $\beta'(t, p_s)$ as described next. Let $X = \{x_1, \ldots, x_n\}$ be the set distinct variables of type λ mentioned in the inscriptions of incoming arcs of t, that is, $X = \{x | x \in \beta(p, t) \text{ for some } p \in P\}$. These variables denote the n distinct objects of type λ accessed upon firing t. We then need to check for the presence of the n distinct assigned resources to this objects, by defining $\beta'(p_s, t) = \beta'(t, p_s) = (x_1, r_1) + \ldots + (x_n, r_n)$, where $r_1, \ldots, r_n$ are n distinct (resource) variables of type η.

A t-PNID is called a *(full) resource closure of* N if it is obtained by recursively constructing, starting from N, λ_i-resource closures for every $\lambda_i \in type_\Lambda(N)$.

The closure(s) of a marked t-PNID is defined by closing the t-PNID as per Definition 6.5, and enriching the initial marking by populating the resource places with some resources.

Definition 6.6. Let (N, m_0) and (N', m_0') be two marked t-PNIDs, and $\lambda \in type_\Lambda(N)$ be an object type. We say that (N', m_0') is a λ-*marked resource closure* of N if the following conditions hold:

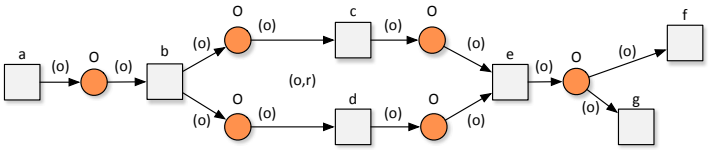
1. N' is a λ-resource closure of N;
2. m_0' extends m_0 by assigning to the resource place p_r introduced in N' a *finite subset* of identifiers of type $type(p_r)$.

A marked t-PNID is called a *(full) marked resource closure of* (N, m_0) if it is obtained by recursively constructing, starting from (N, m_0), λ_i-marked resource closures for every $\lambda_i \in type_\Lambda(N)$.

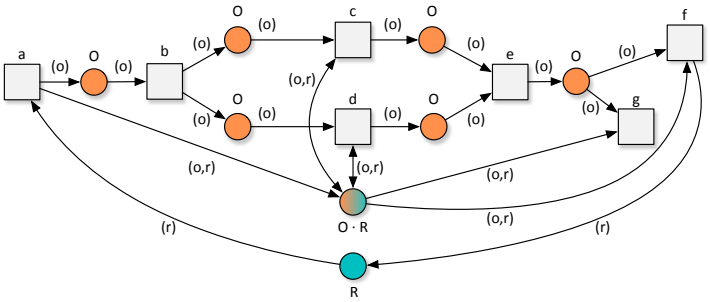
Notice that, in Definition 6.6, we obey to the named approach described in the opening of this section by assigning to the resource place a set (and not a multi-set) of resources.

It is easy to see that, for a t-PNID N and a type λ, there are only finitely many distinct λ-closures of N, obtained by choosing which λ-consumers actually return resources in the resource place of the closure, and which internal λ-transitions access the assignment place of the closure. Marked t-PNIDs defined on these finitely many distinct closures only differ by the set of identifiers they initially assign to each resource place.

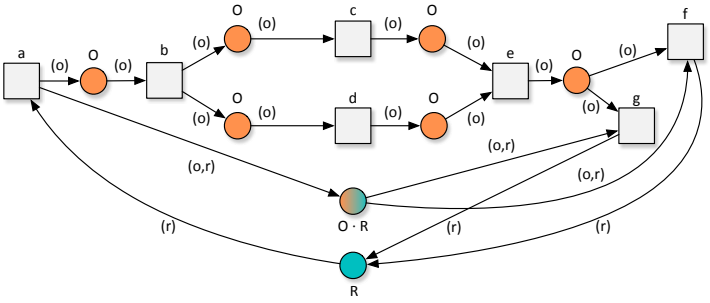
Example 6.7. Figure (11c) shows an (order-)resource closure for the t-PNID manipulating orders via the *create order* and *send order* transitions, and the orange place in between. In particular, the resource type of clerk is chosen for the closure. The green place (in fact, typed *clerk*) at the bottom of the picture is the resource place of the closure, while the mixed-colored place (in fact, typed *order·clerk*) is the assignment place of the closure. The inscriptions attached to the input/output arcs connecting these two places to the emitter and consumer transitions for orders indicate how identifiers are being matched when consuming/returning resources, while recalling the objects they get assigned to.



(a) A sound t-PNID manipulating an object type $O \in \Lambda^o$



(b) An O-resource closure of the t-PNID (a)



(c) Another O-resource closure of the t-PNID (a)

Figure 12: An identifier-sound t-PNID and two alternative resource closures

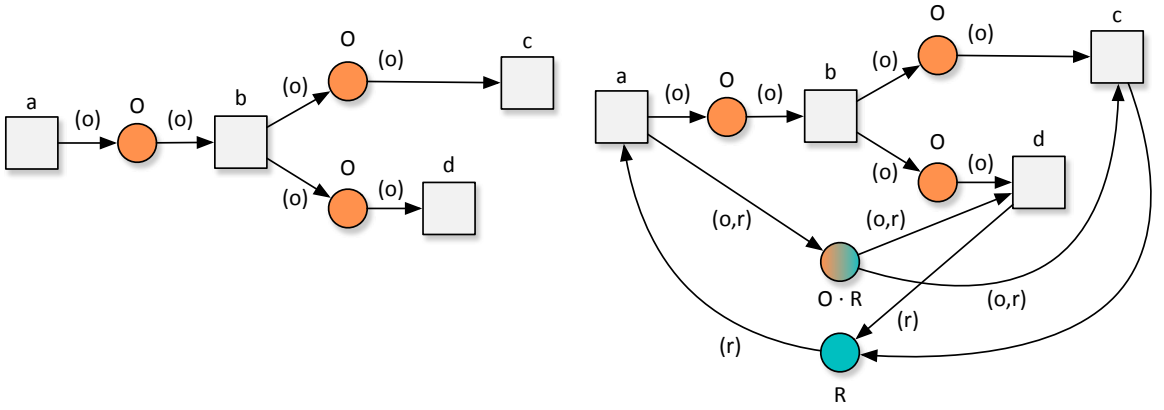
Example 6.8. Figure (12a) illustrates a t-PNID with a single object type O, manipulated using sequential and concurrent transitions, and finally consumed through one of two (mutually exclusive) consumers. Two resource closures of this net are shown in Figures (12b) and (12c), using R as resource type.

The closure in Figure (12b) shows that for an object o that is consumed by transition f, its assigned resource is returned to the resource place, becoming again available for a further assignment. If o is instead consumed by transition g, the resource is also consumed (fetching it from the assignment place without returning it to the resource place). In addition, the two transitions c and d, which are concurrent in the original t-PNID of Figure (12a), are now declared to require the resource *in isolation*, therefore implicitly requiring serialization (in whatever order).

The closure in Figure (12c) depicts a slightly different scenario. On the one hand, both consumers now return the assigned resource upon consuming an object. On the other hand, no internal transition

of the original t-PNID are linked to the assignment place, hence transitions c and d continue to be truly concurrent even after the application of the resource closure.

Examples 6.7 and 6.8 show different examples of well-behaved resource closures, which indeed technically substantiate the informal modelling principles listed above and, even more, actually satisfy the property of conservative resource management, as per Definition 6.2. A natural question is whether this holds when resource closure is applied to an arbitrary input t-PNID. It is easy to show on even very minimalistic examples that provide a negative answer to this question.



(a) An unsound t-PNID with an object type $O \in \Lambda^o$ (b) An O-resource closure of the t-PNID (a)

Figure 13: An identifier-unsound t-PNID and one of its resource closures

Example 6.9. Figure 13 illustrates a t-PNID and one of its resource closures. One can immediately see that the resource closure does not behave as expected. Upon creation of a new object of type O, say, o_1, a resource, say, r_5 of type R is taken from the resource place and assigned to o_1, keeping track of the assignment (o_1, r_5) in the assignment place. Object o_1 then flows through the t-PNID, leading to the generation of two tokens carrying o_1, concurrently enabling the consumer transitions c and d. Upon the consumption of the first of such two tokens, the assignment token (o_1, r_5) is removed from the assignment place and used to return r_5 to the resource place. This means that the second token carrying o_1 stays forever stuck, and cannot be consumed, as there is no assignment token matching o_1 that can be used to fire the consumer transition.

By more closely inspecting the negative example of resource closures discussed in Example 6.9, one can notice that the modelling glitch is not natively caused by the resource closure itself, but actually originates from the fact that the input t-PNID of Figure (13a) is not identifier-sound. In particular, considering this t-PNID, initially marked by the empty marking (or, equivalently, the t-PNID of Figure (13b), initially marked by a marking that inserts some resources in the resource place), the violated property is the one of proper completion: once one of the two concurrent consumer transitions c and d is fired to consume a previously created object, the identifier of the object still persists in the token enabling the other consumer transition. The resource closure actually inherits the lack of proper completion, but also suffers of lack of weak termination.

This leads to the following, natural follow-up question: how does identifier-soundness of an arbitrary input t-PNID impact on the properties of the t-PNIDs resulting from the application of resource closure? We answer by showing three key properties:

1. *Resource closure guarantees conservative resource management* - the application of resource closure to a t-PNID leads to a t-PNID that indeed satisfies the property of conservative resource management.
2. *Resource closure preserves soundness* - Every resource closure of an identifier-sound t-PNID is an identifier-sound t-PNID.
3. *Resource closure of a sound net induces boundedness* - Every resource closure of an identifier-sound t-PNID is a bounded t-PNID.

We start with conservative resource management.

Theorem 6.10. Every marked resource closure of a marked t-PNID manages resources conservatively, in the sense of Definition 6.2.

Proof:
Consider a marked t-PNID (N, m_0), and a marked resource closure (N', m'_0) of it. Fix an object type λ. First, notice that the resource type associated to λ does not have any emitter transition. This proves resource preservation.

We then consider resource exclusive assignment. By definition, in m'_0, every active λ-resource r is referenced by a single token carrying the unary tuple (r) in the resource place for λ. The content of this place is left unaltered until a λ-emitter transition fires. Consider now the firing a λ-emitter generating a new λ object, say o. As stated in Definition 6.5 (items 3a, 3c, and 4b), this can only occur if there is a resource, say, r, in the resource place, and firing leads to consume the only token carrying r therein, while producing a single token carrying (o, r) in the assignment place for λ. Internal λ-transitions executed for o may only access such a token in a read-only mode, thus leaving the content of the assignment place unaltered, as stated in Definition 6.5 (items 3e, and 4d). The only way of removing such a token (o, r) is by firing a λ-collector transition, which, according to Definition 6.5 (items 3b, 3d, and 4c), consumes (o, r) and can possibly produce a token carrying r, inserted in the resource place for λ. All in all, for every resource r associated to the resource place for λ in M'_0, and for every reachable marking $M \in \mathcal{R}(N', M'_0)$, we have that there exists at most one token in M either carrying (r) (and contained in the resource place for λ) or (o, r) for some object o of type λ (and contained in the assignment place for λ). This proves that (N', m'_0) satisfies the property of resource exclusive assignment. □

We continue with soundness preservation. The crux here is that the enrichment with a t-PNID with resources through resource closure does not alter the evolution of emitted objects, but only constrains *when* new objects can be created.

Theorem 6.11. Let (N, m_0) be a marked t-PNID, and (N', m'_0) one of its marked resource closures. If (N, m_0) is identifier-sound, then (N', m'_0) is identifier-sound as well.

Proof:
By definition of resource closure, the reachability graph $\mathcal{R}(N', m'_0)$, projected on the original places of N, is a subset of $\mathcal{R}(N, m_0)$. In fact, the only effect of resource closure on the original marked net is to prevent the possibility of creating new objects if the resource place attached to the corresponding emitter is empty. Since proper termination is a universal property over markings, it is preserved by subsets of $\mathcal{R}(N, m_0)$ and, noticing that resource and assignment places do not affect the status of proper completion, hence also by $\mathcal{R}(N', m'_0)$. This proves that (N', m'_0) properly completes.

As for weak termination of $\mathcal{R}(N', m'_0)$, assume by absurdum that one object o of type λ cannot progress to consumption. Since the original t-PNID is weakly terminating, this can only happen due to the absence of (the only) assignment token referencing o and being present in the assignment place associated to λ. Such a token was by construction generated upon the creation of o, and consequently its absence can be only due to a previous firing of some collector transition that consumed a token referencing o. This would however mean that $\mathcal{R}(N', m'_0)$ is not properly completing, which contradicts what was proven before. □

Finally, we turn to boundedness of resource closures of sound t-PNIDs. Intuitively, this is due to the good interaction between resources and objects:

- there are boundedly many resources available;
- new objects can be created as long as there are still available resources;
- when no resource is available, a new object can be created only upon destruction of a currently existing one.

This reconstructs, in the more sophisticated case of t-PNIDs, the boundedness result for resource and instance-aware workflow nets, at the core of [20].

Theorem 6.12. Let (N, m_0) be a marked t-PNID, and (N', m'_0) one of its marked resource closures. If (N, m_0) is identifier-sound, then (N', m'_0) is bounded.

Proof:
Boundedness immediately holds for resource identifiers in $\mathcal{R}(N', m'_0)$, thanks to Theorem 6.10 and the fact that (N', m'_0) manages resources conservatively, and to the fact that, by construction of resource closure, every resource originally present in m'_0 is either carried by one token present in its corresponding resource place, or by one token present in its corresponding assignment place.

We then prove the theorem by showing that (N', m'_0) is width- and depth-bounded when considering object identifiers.

Depth boundedness is immediately obtained by recalling that, by Theorem 6.11, identifier-soundness of (N, m_0) implies identifier-soundness of (N', m'_0), which is consequently depth-bounded by Lemma 4.9.

Width-boundedness instead follows from the observation, already used in the proof of Theorem 6.10, that for every marking $m \in \mathcal{R}(N', m'_0)$, if $o \in Id(m)$, then there is exactly one token referencing o in the assignment place associated to the type of o. Hence the maximum number of object identifiers is bounded by the number of resources present in m'_0. □

By combining Theorems 6.10 and 4.20, we thus obtain the following conclusive result, that relates to the verification results discussed in Section 4.3.

Corollary 6.13. Model checking of $\mu\mathcal{L}\text{-}FO^{\text{PNID}}$ and $LTL\text{-}FO_p^{\text{PNID}}$ is decidable for marked resource closures of identifier-sound t-PNIDs.

6.2. Discussion

We now briefly comment on the interestingness and generality of our approach to resource management. First and foremost, the conservative management of resources is a quite general notion, which is for example naturally guaranteed in processes operating over material objects (called *material handling systems* in [34]). In every snapshot of such system, each object can either be under the responsibility of one and only one resource (e.g., a baggage being inspected by an operator), or moving from one resource to another (e.g., a baggage waiting for inspection in a queue). In this light, resource closure captures the prototypical case where an object is associated to a single resource alongside its entire lifecycle. While taking verbatim this notion is hence unnecessarily restrictive, it allowed us to highlight the essential features needed to suitably control the way a t-PNID interacts with resources:

1. resources should be managed conservatively;
2. at every point in time, resources should control the number of simultaneously active objects of each type (so that a new object can be created only if there is an available resource to assign to it).

The latter requirement can be achieved by more fine-grained notions of resource closures, applied to sub-nets operating over the same object type. For example, if an object type is manipulated via two subnets that are composed sequentially, concurrently, or in mutual exclusion, one may apply resource closure over each subnet (possibly using distinct resource types), while retaining all the good properties introduced here. This calls for a follow-up investigation: infusing resource closure within the constitutive blocks of typed Jackson nets, ensuring that each block operates over a dedicated resource type satisfying the two features 1. and 2. recalled above.

Last but not least, such two features yield the key property that the resulting net is bounded, as stated in Theorem 6.12. This should by no means be interpreted as the fact that the t-PNID overall operates over boundedly many objects: in fact, unboundedly many objects can be created and handled, provided that they are not all simultaneously active within the system, but distribute over time depending on the amount of available resources.

7. Related work

This work belongs to the line of research that aims at augmenting pure control-flow description of processes with data, and study formal properties of the resulting, integrated models. When doing so, it becomes natural to move from case-centric process models whose analysis focuses on the evolution of a single instance in isolation, to so-called *object-centric process models* where multiple related instances of the same or different processes co-evolve. This is relevant for process modeling, analysis, and mining [35].

Different approaches to capture the control-flow backbone of object-oriented processes have been studied in literature, including declarative [36] and database-centric models [37]. In this work, we follow the Petri net tradition, which comes with three different strategies to tackle object-centric processes.

A first strategy is to represent objects implicitly. The most prominent example in this vein is constituted by proclets [11]. Here, each object type comes with a Petri net specifying its life cycle. Special ports, annotated with multiplicity constraints, are used to express generation and synchronization points in the process, operating over tokens that are implicitly related to co-referring objects. Correctness analysis of proclets is an open research topic.

A second strategy is to represent objects explicitly. Models adopting this strategy are typically extensions of ν-PNs [12], building on their ability to generate (fresh) object identifiers and express guarded transitions relating multiple objects at once. While ν-PNs attach a single object to each token, Petri nets with identifiers (PNIDs) [2] use vectors of identifiers on tokens, representing database transactions. Representing and evolving relationships between objects call for extending this to tuples of objects, in the style of [2]. For such Petri nets with identifiers (PNIDs), [2] provides patterns capturing different types of database transactions. The ISML approach [6] equips Petri nets with identifiers (PNIDs) [2] with the ability of manipulating populations of objects defining the extensional level of an ORM data model. Transitions can be executed if they do not lead to violating the constraints captured in the data model. For such models, correctness properties are assessed by imposing that the overall set of object identifiers is finite, and fixed a-priori. This ensures that the overall state space is indeed finite, and can be analyzed using conventional methods. Catalog-nets [38] extend PNIDs with the ability of querying a read-only database containing background information. Correctness properties are checked parametrically to the content of read-only database. Decidability and other meta-properties, as well as actual algorithms for verification based on SMT model-checking, are given for safety properties, whereas (data-aware) soundness can only be assessed for state-bounded systems [39, 37].

The third, final strategy for modeling object-centric processes with Petri nets is to rely on models that highlight how multiple objects of different types may flow through shared transitions, without considering object identifier values. This approach is followed in [40], where object-centric nets are extracted from event logs, where logged events might come with sets of object identifiers. Soundness for this model is studied in [41], where the authors propose to check related correctness criteria for object types, without considering concrete object identifiers, or for single objects in isolation that are still allowed to interact with the system environment along their life-cycles. Similarly to the approach studied in this paper, the authors in [41] assume that the system model can have any number of objects being simultaneously active.

The approach studied in this paper focuses on the essence of Petri net-based object-centric processes adopting the explicit approach, that is, grounded on PNIDs. We provide, for the first time, a notion of *identifier soundness* that conceptually captures the intended evolution of objects within a net, show that such a property is undecidable to check in general, and provide a pattern-based construction technique that guarantees to produce identifier-sound models. Other works that propose more high-level extensions of classical WF-nets as well as the related notion(s) of soundness. In [42, 43, 44], the

authors investigated data-aware soundness for data Petri nets, in which a net is extended with guards manipulating a finite set of variables associated with the net. That type of soundness was shown to be decidable. In [20], the authors proposed both a workflow variant of ν-Petri nets and its resource-aware extension. The authors also defined a suitable notion of soundness for such nets and demonstrated that it is decidable by reducing the soundness checking problem to a verification task over another first order logic-based formalism. [45, 46] considered the soundness property for BPMN process models with data objects that can be related to multiple cases. The approach consists of several transformation steps: from BPMN to a colored Petri net and then to a resource-constrained workflow net. The authors then check k-soundness against the latter.

Resource-constrained workflow nets pose different requirements on soundness. In [47] the authors studied a specific class of WFR-nets for which soundness was shown to be decidable. In [48, 49] a more general class of Resource-Constrained Workflow Nets (RCWF-nets) was defined. The constraints are imposed on resources and require that all resources that are initially available are present again after all cases terminate, and that for any reachable marking, the number of available resources does not override the number of initially available resources. In [48] it was proven that for RCWF-nets with a single resource type generalized soundness can be effectively checked in polynomial time. Decidability of generalized soundness for RCWF-nets with an arbitrary number of resource places was shown in [49].

8. Conclusions

Achieving harmony in models that describe how processes data objects manipulate is challenging. In this paper, we use typed Petri nets with Identifiers (t-PNIDs) to model these complex interactions of multiple objects, referred through their identifiers. We propose identifier soundness as a correctness criterion that conceptually captures the expected evolution of each object. Identifier soundness consists of two conditions: weak termination, i.e., that any identifier that is created is eventually removed, and proper type completion, i.e., when a collecting transition fires for an identifier of a type, the type should be removed from the resulting marking. Identifier soundness is in general undecidable for t-PNIDs. For two subclasses we show that identifier soundness is guaranteed, and that the overall model remains live. On top of that, we propose a resource-aware extension of t-PNIDs, in which all object manipulations are systematically guarded by a finite number of typed resources. For this class of t-PNIDs we propose a correctness criterion similar to identifier soundness, which also takes into account conditions for correct resource management. The resource soundness is deemed to be undecidable as well.

Many systems allow for a dynamic number of simultaneously active objects. In theory, this number can be infinite, and thus such models become width-unbounded. However, for many systems there is a natural upper bound, which can be either assumed or guaranteed with different modeling techniques (such as multiplicity upper bounds on objects [37] or resources [26, 49]. This gives potential for different directions on new analysis techniques. As an example, some models may have a minimum bound such that its correct behavior is guaranteed above this bound, in a similar way as 1-soundness of WF-nets guarantees correctness of its EC-closure. One can extend t-PNIDs by enriching

objects with attributes over different datatypes, and transitions with the ability to query such attributes and express conditions and updates over them, using their datatype-specific predicates. Of particular interest are comparisons and arithmetic operations for numerical datatypes. This calls for combining the techniques studied in this paper with data abstraction techniques used to deal with numerical datatypes, possibly equipped with arithmetics [43, 50].

We plan to provide tool support for the designer of such systems. Although many correctness criteria are undecidable, designers should be left in the dark. Since the ISM-suite [7] already allows to model t-PNIDs, we intend to work on extending it with verification techniques to support the modeler in designing systems where processes and data are in resonance.

References

[1] Reisig W. Understanding Petri Nets – Modeling Techniques, Analysis Methods, Case Studies. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-33277-7. doi:10.1007/978-3-642-33278-4.

[2] van Hee KM, Sidorova N, Voorhoeve M, van der Werf JMEM. Generation of Database Transactions with Petri Nets. *Fundam. Inform.*, 2009. **93**(1-3):171–184.

[3] Karp RM, Miller RE. Parallel Program Schemata. *J. Comput. Syst. Sci.*, 1969. **3**(2):147–195. doi:10.1016/S0022-0000(69)80011-5.

[4] Hack M. The Recursive Equivalence of the Reachability Problem and the Liveness Problem for Petri Nets and Vector Addition Systems. In: Proc. of SWAT 1974. IEEE Computer Society, 1974 pp. 156–164. doi:10.1109/SWAT.1974.28.

[5] van der Aalst WMP. Verification of Workflow Nets. In: Proc. of Petri Nets, volume 1248 of *LNCS*. Springer, 1997 pp. 407–426. doi:10.1007/3-540-63139-9_48.

[6] Polyvyanyy A, van der Werf JMEM, Overbeek S, Brouwers R. Information Systems Modeling: Language, Verification, and Tool Support. In: Proc. of CAiSE 2019, volume 11483 of *LNCS*. Springer, 2019 pp. 194–212. doi:10.1007/978-3-030-21290-2_13.

[7] van der Werf JMEM, Polyvyanyy A. The Information Systems Modeling Suite - Modeling the Interplay Between Information and Processes. In: Proc. of Petri Nets 2020, volume 12152 of *LNCS*. Springer, 2020 pp. 414–425. doi:10.1007/978-3-030-51831-8_22.

[8] van der Werf JMEM, Polyvyanyy A. An Assignment on Information System Modeling - On Teaching Data and Process Integration. In: Business Process Management Workshops, volume 342 of *LNBIP*. Springer, 2018 pp. 553–566. doi:10.1007/978-3-030-11641-5_44.

[9] Glabbeek R. The Linear Time - Branching Time Spectrum II: The Semantics of Sequential Systems with Silent Moves. In: CONCUR 1993, volume 715 of *LNCS*. Springer, 1993 pp. 66–81.

[10] van Hee KM, Sidorova N, Voorhoeve M. Soundness and Separability of Workflow Nets in the Stepwise Refinement Approach. In: Proc. of Petri Nets 2003, volume 2679 of *LNCS*. Springer, 2003 pp. 337–356. doi:10.1007/3-540-44919-1_22.

[11] Fahland D. Describing Behavior of Processes with Many-to-Many Interactions. In: Proc. of Petri Nets 2019. Springer, 2019 pp. 3–24. doi:10.1007/978-3-030-21571-2_1.

[12] Rosa-Velardo F, de Frutos-Escrig D. Decidability and complexity of Petri nets with unordered data. *Theor. Comput. Sci.*, 2011. **412**(34):4439–4451. doi:10.1016/j.tcs.2011.05.007.

[13] Rosa-Velardo F, Alonso OM, de Frutos-Escrig D. Mobile Synchronizing Petri Nets: A Choreographic Approach for Coordination in Ubiquitous Systems. *Electron. Notes Theor. Comput. Sci.*, 2006. **150**(1):103–126. doi:10.1016/j.entcs.2005.12.026. URL `https://doi.org/10.1016/j.entcs.2005.12.026`.

[14] Rosa-Velardo F, de Frutos-Escrig D. Forward Analysis for Petri Nets with Name Creation. In: Applications and Theory of Petri Nets, volume 6128 of *LNCS*. Springer, Berlin, 2010 pp. 185–205. doi:10.1007/978-3-642-13675-7_12.

[15] van Hee KM, Sidorova N, van der Werf JMEM. Business Process Modeling Using Petri Nets. *Trans. Petri Nets Other Model. Concurr.*, 2013. **7**:116–161. doi:10.1007/978-3-642-38143-0_4.

[16] Lasota S. Decidability Border for Petri Nets with Data: WQO Dichotomy Conjecture. In: Kordon F, Moldt D (eds.), Proc. of Petri Nets 2016, volume 9698 of *LNCS*. Springer, 2016 pp. 20–36. doi:10.1007/978-3-319-39086-4_3.

[17] Ghilardi S, Gianola A, Montali M, Rivkin A. Petri net-based object-centric processes with read-only data. *Inf. Syst.*, 2022. **107**:102011. doi:10.1016/j.is.2022.102011.

[18] Lasota S. Decidability Border for Petri Nets with Data: WQO Dichotomy Conjecture. In: Kordon F, Moldt D (eds.), Application and Theory of Petri Nets and Concurrency - 37th International Conference, PETRI NETS 2016, Toruń, Poland, June 19-24, 2016. Proceedings, volume 9698 of *LNCS*. Springer, 2016 pp. 20–36. doi:10.1007/978-3-319-39086-4_3.

[19] Martos-Salgado M, Rosa-Velardo F. Dynamic Soundness in Resource-Constrained Workflow Nets. In: Formal Techniques for Distributed Systems. FMOODS FORTE, volume 6722 of *LNCS*. Springer, 2011 pp. 259–273. doi:10.1007/978-3-642-21461-5_17.

[20] Montali M, Rivkin A. Model checking Petri nets with names using data-centric dynamic systems. *Formal Aspects Comput.*, 2016. **28**(4):615–641. doi:10.1007/s00165-016-0370-6.

[21] Calvanese D, Giacomo GD, Montali M, Patrizi F. First-order μ-calculus over generic transition systems and applications to the situation calculus. *Inf. Comput.*, 2018. **259**(3):328–347. doi:10.1016/j.ic.2017.08.007.

[22] Calvanese D, Giacomo GD, Montali M, Patrizi F. Verification and Monitoring for First-Order LTL with Persistence-Preserving Quantification over Finite and Infinite Traces. In: Raedt LD (ed.), Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022. ijcai.org, 2022 pp. 2553–2560. doi:10.24963/ijcai.2022/354.

[23] Hariri BB, Calvanese D, Giacomo GD, Deutsch A, Montali M. Verification of relational data-centric dynamic systems with external services. In: Hull R, Fan W (eds.), Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2013, New York, NY, USA - June 22 - 27, 2013. ACM, 2013 pp. 163–174. doi:10.1145/2463664.2465221.

[24] Abiteboul S, Hull R, Vianu V. Foundations of Databases. Addison-Wesley, 1995. ISBN 0-201-53771-0.

[25] van Hee KM, Oanea O, Post R, Somers LJ, van der Werf JMEM. Yasper: a tool for workflow modeling and analysis. In: Proc. of ACSD 2006. IEEE, 2006 pp. 279–282. doi:10.1109/ACSD.2006.37.

[26] Montali M, Rivkin A. Model checking Petri nets with names using data-centric dynamic systems. *Formal Aspects Comput.*, 2016. **28**(4):615–641. doi:10.1007/s00165-016-0370-6. URL `https://doi.org/10.1007/s00165-016-0370-6`.

[27] Leemans SJJ, Fahland D, van der Aalst WMP. Discovering Block-Structured Process Models from Event Logs - A Constructive Approach. In: Proc. of Petri Nets 2013, volume 7927 of *LNCS*. Springer, 2013 pp. 311–329. doi:10.1007/978-3-642-38697-8_17.

[28] Weidlich M, Polyvyanyy A, Mendling J, Weske M. Causal Behavioural Profiles - Efficient Computation, Applications, and Evaluation. *Fundam. Informaticae*, 2011. **113**(3-4):399–435. doi:10.3233/FI-2011-614.

[29] van Hee KM, Hidders J, Houben GJ, Paredaens J, Thiran P. On the relationship between workflow models and document types. *Information Systems*, 2009. **34**(1):178–208. doi:10.1016/j.is.2008.06.003.

[30] Murata T. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 1989. **77**(4):541–580.

[31] Berthelot G. Verification de Reseaux de Petri. Ph.D. thesis, Université Pierre et Marie Curie (Paris), 1978.

[32] van Hee KM, Sidorova N, Voorhoeve M. Resource-Constrained Workflow Nets. *Fundam. Inform.*, 2006. **71**(2-3):243–257.

[33] Lomazova IA, Bashkin VA, Jancar P. Resource Bisimilarity in Petri Nets is Decidable. *Fundam. Informaticae*, 2022. **186**(1-4):175–194. doi:10.3233/FI-222125.

[34] Fahland D, Denisov V, van der Aalst WMP. Inferring Unobserved Events in Systems with Shared Resources and Queues. *Fundam. Informaticae*, 2021. **183**(3-4):203–242. doi:10.3233/FI-2021-2087.

[35] van der Aalst WMP. Object-Centric Process Mining: Dealing with Divergence and Convergence in Event Data. In: Ölveczky PC, Salaün G (eds.), Proc. of SEFM 2019, volume 11724 of *Lecture Notes in Computer Science*. Springer, 2019 pp. 3–25. doi:10.1007/978-3-030-30446-1_1.

[36] Artale A, Kovtunova A, Montali M, van der Aalst WMP. Modeling and Reasoning over Declarative Data-Aware Processes with Object-Centric Behavioral Constraints. In: Proc. of BPM 2019, volume 11675 of *LNCS*. Springer, 2019 pp. 139–156. doi:10.1007/978-3-030-26619-6_11.

[37] Montali M, Calvanese D. Soundness of Data-Aware, Case-Centric Processes. *Int. Journal on Software Tools for Technology Transfer*, 2016. doi:10.1007/s10009-016-0417-2.

[38] Ghilardi S, Gianola A, Montali M, Rivkin A. Petri Nets with Parameterised Data - Modelling and Verification. In: Fahland D, Ghidini C, Becker J, Dumas M (eds.), BPM'20, volume 12168 of *LNCS*. Springer, 2020 pp. 55–74. doi:10.1007/978-3-030-58666-9_4.

[39] Bagheri Hariri B, Calvanese D, Montali M, Deutsch A. State-Boundedness in Data-Aware Dynamic Systems. In: Proc. of KR 2014. AAAI Press, 2014 .

[40] van der Aalst WMP, Berti A. Discovering Object-centric Petri Nets. *Fundam. Informaticae*, 2020. **175**(1-4):1–40. doi:10.3233/FI-2020-1946.

[41] Lomazova IA, Mitsyuk AA, Rivkin A. Soundness in Object-centric Workflow Petri Nets. *CoRR*, 2021. **abs/2112.14994**.

[42] de Leoni M, Felli P, Montali M. A Holistic Approach for Soundness Verification of Decision-Aware Process Models. In: ER, volume 11157 of *LNCS*. Springer, 2018 pp. 219–235. doi:10.1007/978-3-030-00847-5_17.

[43] Felli P, de Leoni M, Montali M. Soundness Verification of Decision-Aware Process Models with Variable-to-Variable Conditions. In: Proc. of ACSD 2019. IEEE, 2019 pp. 82–91. doi:10.1109/ACSD.2019.00013.

[44] de Leoni M, Felli P, Montali M. Strategy Synthesis for Data-Aware Dynamic Systems with Multiple Actors. In: KR. 2020 pp. 315–325. doi:10.24963/kr.2020/32.

[45] Haarmann S, Weske M. Cross-Case Data Objects in Business Processes: Semantics and Analysis. In: BPM (Forum), volume 392 of *Lecture Notes in Business Information Processing*. Springer, 2020 pp. 3–17. doi:10.1007/978-3-030-58638-6_1.

[46] Haarmann S, Montali M, Weske M. Technical Report: Refining Case Models Using Cardinality Constraints. *CoRR*, 2020. **abs/2012.02245**.

[47] Barkaoui K, Benayed R, Sbai Z. Workflow Soundness Verification Based on Structure Theory of Petri Nets. *International Journal of Computing and Information Sciences (IJCIS)*, 2007. **5**:51–62.

[48] van Hee KM, Serebrenik A, Sidorova N, Voorhoeve M. Soundness of Resource-Constrained Workflow Nets. In: ICATPN, volume 3536 of *LNCS*. Springer, 2005 pp. 250–267. doi:10.1007/11494744_15.

[49] Sidorova N, Stahl C. Soundness for Resource-Constrained Workflow Nets Is Decidable. *IEEE Trans. Syst. Man Cybern. Syst.*, 2013. **43**(3):724–729. doi:10.1109/TSMCA.2012.2210415.

[50] Felli P, Montali M, Winkler S. Linear-Time Verification of Data-Aware Dynamic Systems with Arithmetic. In: Proc. of AAAI 2022. AAAI Press, 2022 .