A component framework where port compatibility implies weak termination

Citation for published version (APA):

Document status and date:
Published: 01/01/2011

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.
A Component Framework where Port Compatibility Implies Weak Termination

Debjyoti Bera, Kees M. van Hee, Michiel van Osch, and
Jan Martijn van der Werf

Department of Mathematics and Computer Science,
Technische Universiteit Eindhoven,
P.O. Box 513, 5600 MB Eindhoven, The Netherlands.
{d.bera, k.m.v.hee, m.p.w.j.van.osch, j.m.e.m.v.d.werf}@tue.nl

Abstract. The design and verification of an asynchronous communicating system can be very complex. In this paper we focus on weak termination: in each reachable state, the system has the option to eventually terminate. We present a component framework and construction method that guarantees weak termination. In the framework, communication between components is modeled by portnets, a special class of workflow nets. A basic component defines the orchestration of the portnets. For weak termination, the orchestration should accord to each of the portnets. A composite component is built from basic components that offer some service via a portnet. We provide sufficient conditions to guarantee weak termination for composite components. Furthermore, we present a refinement-based construction procedure to derive a weakly terminating composite from an architectural diagram of the system.

1 Introduction

The class of asynchronous communicating systems encompasses a wide range of software systems that include information systems, embedded systems, grid computing systems, etc. The distributed nature and growing complexity of these systems warrant the need for a component based development (CBD) approach with support for formal analysis techniques. The central idea in the design of such systems involves the construction of complex systems by assembling components while guaranteeing certain properties.

Over the past years, different formal models supporting component based development have been proposed, like Cadena [7] and SaveCCM [4]. Many of these techniques provide a model to specify the components and their composition while relying on state space based explorations to verify the correctness of the design. State space based explorations are generally time consuming and do not scale well to the complexities of real world models. For this reason we construct a framework to guarantee correctness properties by construction. We focus on one property: weak termination.

The weak termination property states that in each reachable state of the system, the system always has the possibility to reach a final state. Generalized
soundness [9] is a generalization of weak termination for workflow nets. A class of
generalized sound workflow nets is the class of ST-nets [9] which are constructed
by successive refinements of state machines and acyclic marked graphs [5].

Components are loosely coupled. As a consequence, their composition in-
trudes a high degree of concurrency, and thus a state space explosion. In [2]
a sufficient condition is presented to pairwise verify weak termination for tree
structured compositions. For a subclass of compositions of pairs of components,
called ATIS-nets, this condition is implied by their structure [10]. ATIS-nets are
constructed from pairs of acyclic marked graphs and isomorphic state machines,
and the simultaneous refinement of pairs of places [14].

In this paper, we present a component framework to construct a network of
asynchronously communicating components that guarantees weak termination.
The framework supports a best practice in communication protocol design: com-
munication between two components is first modeled as a state machine. Then,
each transition is assigned to one of the components, such that if any two tran-
sitions are in conflict, these transitions are designated to the same component.
Then, the state machine is duplicated for each of the components. If a transition
is assigned to that component, it sends a message; the corresponding transition
of the other component receives this message. Such a net is called a portnet. In
this way, a component consists of a set of portnets defining its behavior with
the environment. A component needs to orchestrate all its portnets, such that
for each component it communicates with, it acts as the corresponding portnet.
This requirement is similar to the condition imposed by choreography standards
like WS-CDL [11]. A Component may be either basic or composite. A basic com-
ponent provides a service via a portnet. In order to do so, it consumes from other
components. In a composition, we allow more than one component to consume
a service from another component. Such a composition is a directed graph with
edges representing dependency relationships between basic components. If the
composition is acyclic, it is a composite component.

The orchestration of a component may nest portnets. To resemble this in
the architecture, we introduce a simple architectural diagram. Furthermore, we
study the behavior of an arbitrary composition of components and give sufficient
conditions to guarantee weak termination. We also present a construction proce-
dure based on the rules of [13] and [8] to derive a weakly terminating composition
of components.

2 Preliminaries

Let $S$ be a set. We denote the powerset by $\mathcal{P}(S)$. A bag over some set $S$ is a
function $m : \mathbb{N} \rightarrow S$, where $\mathbb{N} = \{0, 1, 2, \ldots\}$ denotes the set of natural numbers.
For $s \in S$, $m(s)$ denotes the number of occurrences of $s$ in $m$. We enumerate
bags with square brackets, e.g. the bag $m = [a^2, b^3]$ has an element $a$ occurring
twice and element $b$ occurring thrice and all other elements have multiplicity
zero. The set of all bags over $S$ is denoted by $B(S)$. We write $[]$ for an empty
bag and we use $+$ and $-$ for the sum of two bags and $=, <, >, \leq, \geq$ to element
wise compare bags, which are defined in the standard way. A set can be seen as a multiset in which each element of the set occurs exactly once.

A finite sequence $\sigma$ over some set $S$ of length $n \in \mathbb{N}$ is a function $\sigma : \{1...n\} \rightarrow S$. The length of a sequence is denoted by $|\sigma|$. We represent a sequence of length $n$ by $\sigma = \langle s_1, ..., s_n \rangle$ where $s_1, ..., s_n \in S$ and $\sigma(i) = s_i$ for $1 \leq i \leq n$. If $|\sigma| = 0$, it is the empty sequence denoted by $\epsilon$. The concatenation of two finite sequences $\sigma = \langle a_1, ..., a_n \rangle$ and $\sigma' = \langle b_1, ..., b_m \rangle$ is denoted by $\sigma \sigma'$ and is the sequence $\langle a_1, ..., a_n, b_1, ..., b_m \rangle$ of length $n + m$. The Parikh vector of a sequence $\sigma$, denoted by $\vec{\sigma}$, is a bag representing the number of occurrences of each element in $\sigma$. A sequence $\sigma$ is a prefix of a sequence $\gamma$ if a sequence $\sigma'$ exists such that $\sigma \sigma' = \gamma$. The projection of a sequence $\sigma \in S^*$ onto a set $Q$ is defined inductively as $\epsilon_Q = \epsilon; ((s_1) \sigma)_Q = ((s_1)) \sigma_Q$ if $s_1 \in Q$ and $((s_1) \sigma)_Q = \sigma_Q$ if $s_1 \notin Q$. We denote the set obtained by interleaving two sequences $\sigma$ and $\gamma$ by $\sigma||\gamma$.

**Petri Nets** A Petri net is a tuple $N = (P, T, F)$, where $P$ is the set of places; $T$ is the set of transitions such that $P \cap T = \emptyset$ and $F$ is the flow relation $F \subseteq (P \times T) \cup (T \times P)$. We refer to elements from $P \cup T$ as nodes and elements from $F$ as arcs. We denote the places of net $N$ by $P_N$, transitions as $T_N$ and similarly for other elements of the tuple. If the context is clear, we omit $N$ in the subscript. We define the preset of a node $n$ as $\bullet n = \{m|(m, n) \in F\}$ and the postset as $m\bullet N = \{n|(m, n) \in F\}$. We lift the notion of a preset and postset to sets: $\bullet S = \cup_{s \in S} \bullet s$ and $S\bullet = \cup_{s \in S} s\bullet$ for some set $S \subseteq (P \cup T)$. If the context is clear, the subscript is omitted. A path $\nu \in (P \cup T)^*$ in a Petri net $N$ of length $n \in \mathbb{N}$ is a sequence such that $(\nu(i), \nu(i + 1)) \in F$ for all $1 \leq i < n$. We denote a path of length $n$ by $\nu = \langle x_1, ..., x_n \rangle$ where $x_i = \nu(i)$ for all $q \leq i \leq n$. The set of all paths of a Petri net $N$ is called the *path space* and denoted by $PS(N)$. Two Petri nets $N$ and $M$ are disjoint if $(P_N \cup T_N) \cap (P_M \cup T_M) = \emptyset$. They are *isomorphic*, denoted by $N \cong_M$ if and only if a bijective function $\psi : P_N \cup T_N \rightarrow P_M \cup T_M$ exists such that $P_M = \psi(P_N)$, $T_M = \psi(T_N)$ and $\forall(x, y) \in F_N \Leftrightarrow (\psi(x), \psi(y)) \in F_M$. We write $N \cong M$ if a bijective function $\psi$ exists such that $N \cong_M M$. The state of a Petri net $N = (P, T, F)$ is determined by its marking which represents the distribution of tokens over places of the net. An marking of a Petri net $N$ is a bag over its places $P$, i.e., $m \in B(P)$. A transition $t \in T$ is enabled in $m$ if and only if $t \in m$. An enabled transition may fire which results in a new marking $m' = m \rightarrow t$, denoted by $m \xrightarrow{t} m'$. We define the set of reachable markings of a Petri net $N$ with marking $m$ inductively by $R(m) = \{m\} \cup \bigcup_{m \xrightarrow{t} m'} R(m')$. We lift the notion for firing and enabledness to firing sequences. A firing sequence $\sigma \in T^*$ is enabled in $m$ if and only if $\exists m_1, ..., m_n \in B(P)$ : $m \xrightarrow{t_1} m_1 \xrightarrow{t_2} ... \xrightarrow{t_n} m_n$ and $m_n \xrightarrow{t} m'$. We write $m \xrightarrow{t} m'$ and call $m'$ reachable from $m$, if $m \xrightarrow{t} m'$. We denote the set of all reachable markings from marking $m$ of a Petri net $N$ by $R(N, m) = \{m'|m \xrightarrow{t} m'\}$. Furthermore, the number of occurrences of the transitions belonging to the set $A \subseteq T$ in any arbitrary firing sequence $\sigma \in T^*$ is denoted by $\text{occurs}(\sigma, A) = \sum_{t \in A} \sigma(t)$. We define the *net system* of a Petri
net $N$ as a 3-tuple $M = (N, m_0, m_f)$, where $m_0 \in B(P_N)$ is the initial marking and $m_f \in B(P_N)$ is the final marking. The weak termination property for a net system $M$ states that $\forall m \in R(N, m_0) : m_f \in R(N, m)$, i.e. for all reachable markings from the initial marking the final marking is reachable. If a marking does not enable any transition in the net, it is called a dead marking. A place is called safe in a net system $(N, m_0, m_f)$ if $\forall m \in R(N, m_0), m(p) \leq 1$. Let $N = (P, T, F)$ be a Petri net. Net $N$ is a workflow net (WFN) if there exists exactly one place $i \in P$ with $\bullet i = \emptyset$, called the initial place, one place $f \in P$ with $f^* = \emptyset$, called the final place, and all nodes $n \in P \cup T$ are on a path from $i$ to $f$. The closure of a workflow net $N$ is a net $\text{closure}(N) = (P, T \cup \{t\}, F \cup \{(t, i), (f, t)\})$ such that $t \notin T$ and $\bullet t = f^* = \{f\}$. A WFN $N$ weakly terminates if its net system $(N, [i], [f])$ weakly terminates. Note that in [9] this property is called 1-Soundness. For an overview of soundness, see [1]. Net $N$ is a state machine (S-net) [5] if and only if $\forall t \in T : |\bullet t| = |f^*| = 1$. In a state machine, a place $p$ is called a split if $p^* > 1$. Likewise, it is a join if $p^* > 1$. Net $N$ is a marked graph (T-net) [5] if and only if $\forall p \in P : |\bullet p| = |p^*| = 1$. A workflow net that is also a state machine is called an S-WFN. If it is both a workflow net and a marked graph, it is called a T-WFN. The class of ST-nets were introduced in [9]. These nets allow both concurrency and choice. Note that the class of T-nets used in [9] has transitions as the initial and final nodes of the net. We extend such a net to our definition of a T-WFN by adding one initial and one final place. The class of ST-nets that we will use in this paper includes the class of S-nets, T-nets and nets obtained after arbitrary successive refinement of places [9] within an ST-net by either an S-WFN or a T-WFN.

3 Component Framework

In this section, we introduce a compositional framework to describe component based systems built of components that are cyclic in their execution and react to inputs from their environment. The main concept of this framework is the notion of a component. A component may be basic or composite. A basic component provides a service and may in turn may use the services offered by other basic components. The interfaces of a basic component are modeled as a portnet. A portnet describes a communication protocol. Such a protocol describes all possible sequences of messages that may be exchanged during a service negotiation. A basic component is a closed ST-net providing some service by means of a sell side portnet and consuming services using buy side portnets from components that have a compatible sell side portnet. The sell side portnet of a basic component encapsulates all of its buy side portnets. Furthermore, we allow buy side portnets to be nested. A composite component is the composition of a set of pairwise composable basic components such that their dependency graph is acyclic.

A component is modeled as a Petri net. An activity within such a component is modeled by a transition. We distinguish between two types of places, namely internal places and interface places. An interface place is either an input place
for one component or an output place for another component. An input place has an empty preset and an output place has an empty postset. All other places of a component are referred to as internal places. Tokens residing at interface places represent messages, otherwise they are simply state markers. Transitions are either internal transitions or interface transitions. An internal transition has no interface places in its preset and postset, whereas an interface transition has some interface places in its preset (then it is called a receive transition) or its postset (then it is called a send transition), but never in both.

3.1 Formalization

Our component framework is based on open Petri nets (OPN) which are a subclass of classical Petri nets. OPN are ideal to model communicating systems. This is because they have a distinguished set of interface places that represent the interfaces of the net. A direct consequence of the interaction of an OPN with its environment results in tokens being exchanged between these places. Furthermore, we add structural constraints to derive subclasses of an OPN. A subworkflow net is a OWN that is a subnet of an OPN.

Definition 1 (Open Petri net, subworkflow net). An open Petri net (OPN) is defined as $N = (P, I, O, T, F, i, f)$, where (1) $P$ is the set of internal places; (2) $I$ is the set of input places with $I = \emptyset$; (3) $O$ is the set of output places with $O = \emptyset$; (4) $T$ is the set of transitions; (5) the sets $P$, $I$, $O$ and $T$ are pairwise disjoint; (6) $i \subseteq P$ is the set of initial places; (7) $f \subseteq P$ is the set of final places; (8) $\forall t \in T : \bullet t \cap I \neq \emptyset \Rightarrow \bullet t \cap O = \emptyset \wedge \bullet t \cap O \neq \emptyset \Rightarrow t \cap I = \emptyset$; and $(P \cup I \cup O, T, F, i, f)$ is the net system. We refer to the set $I \cup O$ as the interface places of the net. The skeleton of $N$ is a Petri net defined as skeleton$(N) = (P, T, F')$, where $F' = F \cap ((P \times T) \cup (T \times P))$. The skeleton system is defined as $(P, T, F', i, f)$.

If skeleton$(N)$ is a WFN then $N$ is called an open workflow net (OWN). If skeleton$(N)$ is a S-WFN then $N$ is called a state machine open workflow net (S-OWN). If skeleton$(N)$ is a T-WFN then $N$ is called a marked graph open workflow net (T-OWN). If skeleton$(N)$ is a ST-net then $N$ is called a ST open workflow net (ST-OWN).

Let $N$ be an OPN and $M$ be a OWN. We say that $M$ is a subworkflow net of $N$ denoted by $M \subseteq N$ if and only if $P_M \subseteq P_N$, $T_M \subseteq T_N$, $F_M \subseteq F_N$, $I_M \subseteq I_N$, $O_M \subseteq O_N$, $N(T_M \cup O_M \cup P_M \setminus \{i_M\}) \cup (T_M \cup I_M \cup P_M \setminus \{f_M\}) \subseteq (T_M \cup P_M \cup I_M \cup O_M)$.

The transitions of an open Petri net are distinguished into three categories depending on the direction of communication, namely send, receive and internal. A send transition contains an output place in its postset. A receive transition has an input place in its preset. A transition that does not send or receive is called an internal transition.

Definition 2 (Direction of communication). The direction of communication of a transition with respect to a place in an open Petri net $N$ is a function
We call a transition \( \lambda \) defined as \( \lambda(t) = \text{send} \iff t^\bullet \cap O \neq \emptyset \land t \cap I = \emptyset; \lambda(t) = \text{receive} \iff t^\bullet \cap I \neq \emptyset \land t^\bullet \cap O = \emptyset \) and \( \lambda(t) = \tau \), otherwise, for all \( t \in T \).

We call a transition \( t \in T \) a communicating transition if and only if \( \lambda(t) \neq \tau \).

The refinement of safe places in a Petri net is a well-known refinement step and has been described in various contexts [9]. We present here the refinement of a safe place within an OPN by an ST-OWN. Note that the choice of a place to be refined must not be either the initial or the final place of the OPN. Furthermore, we also present the reduction of an OPN by an OWN. Note that the reduction operation is the dual of the refinement operation.

**Definition 3 (Place refinement and net reduction).** Given an OPN \( N \) and an OWN \( M \) such that \( N \) and \( M \) are disjoint, a safe place \( p \in P_N \setminus \{n \mid i_N(n) = f_N(n) = 0\} \) can be refined by \( M \), resulting in an OPN \( N' = N \circ_p M = (P, I, O, T, F, i, f) \) with \( P = (P_N \setminus \{p\}) \cup P_M \), \( I = I_N \cup I_M \), \( O = O_N \cup O_M \), \( T = T_N \cup T_M \), \( F = (F_N \setminus \{(p \times \{p\}) \cup (\{p\} \times P^*)\} \cup (\class{F}_M \cup \{(p \times \{i_M\}) \cup (\{f_M\} \times P^*)\})) \), \( i = i_N \), \( f = f_N \). We define the reduction of net \( N' \) by the subworkflow net \( M \) by reduce\((N', M) = N \) if and only if \( N' = N \circ_p M \).

An OPN \( N \) is said to be reducible to another open Petri net \( N' \) if and only if successive applications of the reduce operation on net \( N \) results in the net \( N' \). Note that we restrict the definition to reductions only by the class of ST-net, since this is the inherent structure of all nets in this component framework. Note that this relation is a preorder.

**Definition 4 (Reducible nets).** Consider two OPN’s \( N \) and \( N' \). We say \( N \) is reducible to \( N' \) denoted by \( N \rightsquigarrow N' \) if and only if \( N = N' \lor \exists M : M \text{ is a } \text{ST-OWN} \land (M \neq N) \land (M \subseteq N) \land \text{reduce}(N, M) \rightsquigarrow N' \).

Unlike in an OPN, interfaces in our component framework are more than just a set of interface places acting as message buffers. An interface is determined by a Petri net with a distinguished set of interface places, called the portnet. A portnet defines the communication protocol which specifies all acceptable sequences of messages that are permitted to be exchanged over the portnet.

A portnet is an S-OWN with structural constraints on the relation between transitions and interface places and paths through it. In a portnet, each interface place is connected to exactly one transition, and each transition is connected to exactly one interface place. Secondly, a portnet must satisfy the leg property. A path in a portnet is called a leg if it is a path from a split to a join. We also consider the initial place as a split and the final place as a join. The leg property requires every leg in a portnet to have at least two transitions with different directions of communication. Lastly, a portnet must satisfy the choice property, which requires all transitions belonging to the postset of a place to have the same direction of communication.

**Definition 5 (Portnet).** A portnet \( C \) is a S-OWN such that

\[-\forall t \in T : |(t^\bullet \cup t^*) \cap (I \cup O)| = 1, \text{ i.e. every transition is connected to one interface place};\]
Consider a portnet in either its preset or its postset:

- ∀x ∈ \( I \cup O \): \( |x \cup a^*| = 1 \), i.e. every interface place has one transition from a given portnet in either its preset or its postset;

- (Leg property) ∀β = \( \langle p_1, t_1\ldots t_{n-1}, p_n \rangle \) ∈ \( PS(C) \) : \( (|p_1^*| > 1 \lor p_1 = i_N) \land (|p_n^*| > 1 \lor p_n = f_N) \land \exists t', \beta' : \lambda(t) \neq \lambda(t') \).

- (Choice property) ∀\( t_1, t_2 \in T : \bullet t_1 \cap \bullet t_2 \neq \emptyset \Rightarrow \lambda(t_1) = \lambda(t_2) \), i.e. if two transitions share the same place in their preset, they are of the same type.

We distinguish between two types of portnets: A **sell side portnet** advertises a service and needs a startup message and terminates after sending a result message. A **buy side portnet** consumes a service by sending a startup message and terminates after receiving the result message.

**Definition 6 (Portnet types).** Consider a portnet \( C \). We call Portnet \( C \) a sell side portnet denoted by \( sell(C) \) if and only if ∀\( t \in iC^* \) : \( \lambda(t) = receive \land \forall t \in \bullet f_C : \lambda(t) = send \) and we call Portnet \( C \) a buy side portnet denoted by \( buy(C) \) if and only if ∀\( t \in iC^* \) : \( \lambda(t) = send \land \forall t \in \bullet f_C : \lambda(t) = receive \).

Note that \( \neg sell(C) \Leftrightarrow buy(C) \). A **component** is an OPN with a set of portnets. Every communicating transition in a component belongs to a portnet. Furthermore, every portnet of a component is either already a subworkflow net or there exists a subworkflow net that can be reduced to the corresponding portnet. Note that this subworkflow net is a subnet of the component net and can be defined by appropriately labeling the nodes of this subnet such that it is a OWN.

**Definition 7 (Component).** A component is a pair \( (N, \Gamma) \) where \( N \) is an OPN and \( \Gamma \) is a set of portnets, such that:

- ∀\( t \in T_N : \lambda(t) \neq \tau \Rightarrow \exists C \in \Gamma : t \in TC \)
- ∀\( C \in \Gamma : \exists N' : N' \) is an OWN : \( N' \subseteq N \land N' \rightsquigarrow C \)

The set of all sell side portnets of a component is defined as: \( sellside((N, \Gamma)) = \{ C \in \Gamma \mid sell(C) \} \) and the set of all buy side portnets of a component is defined as: \( buyside((N, \Gamma)) = \{ C \in \Gamma \mid buy(C) \} \).

**Lemma 8 (Preservation of weak termination).** Consider two OPN’s \( N \) and \( M \) such that \( N \rightsquigarrow M \). Then \( N \) is weakly terminating if and only if \( M \) is weakly terminating.

**Proof.** Since the weak termination property of an OPN is defined over its skeleton, the nets being discarded by the reduction operation \( N \rightsquigarrow M \) are ST-nets. By Theorem 36, in [9] this class of nets are generalized sound. Furthermore, by Theorem 10 in [9] the reduced net has the same behavior as the original one. □

Portnets of a component may be nested in each other.

**Definition 9 (Nested portnets).** Consider a component \( (N, \Gamma) \) and two portnets \( C_1, C_2 \in \Gamma \). We say portnet \( C_2 \) is nested in portnet \( C_1 \) denoted by \( C_2 \preceq_N C_1 \) if and only if \( \exists M_1, M_2 : M_2 \subseteq M_1 \subseteq N \land M_1 \rightsquigarrow C_1 \land M_2 \rightsquigarrow C_2 \).
Two portnets are said to be \textit{compatible} if their skeletons are isomorphic. Furthermore, the set of input places of one portnet must match the set of output places of the other portnet while preserving the relation with their associated transitions. Note that a portnet is not compatible with itself.

\begin{definition}[Compatible portnets] Portnets $C_1$ and $C_2$ are compatible with respect to some bijective function $\phi : (P_{C_1} \cup T_{C_1} \cup I_{C_1} \cup O_{C_1}) \rightarrow (P_{C_2} \cup T_{C_2} \cup I_{C_2} \cup O_{C_2})$, denoted by $C_1 \cong_\phi C_2$ if and only if:
\begin{itemize}
  \item skeleton$(C_1) \cong_\phi$ skeleton$(C_2)$,
  \item $O_{C_2} = \phi(O_{C_1})$, $I_{C_2} = \phi(I_{C_1})$,
  \item $\forall x \in I_{C_1}, t \in T_{C_1} : (x,t) \in F_{C_1} \iff (\phi(t),\phi(x)) \in F_{C_2}$,
  \item $\forall x \in O_{C_1}, t \in T_{C_1} : (t,x) \in F_{C_1} \iff (\phi(x),\phi(t)) \in F_{C_2}$
\end{itemize}

We write $C_1 \cong C_2$ if a bijective function $\phi$ exists such that $C_1 \cong_\phi C_2$.

Note that if $C_1 \cong C_2$ and $C_2 \cong C_3$ then $C_1 \cong C_3$.

Basic components are the building blocks of this component framework. The Petri net structure of a basic component is modeled as an ST-OWN with a closure transition. A basic component has one sell side portnet by means of which it provides a service. The sell side portnet may have zero or more nested buy side portnets. Furthermore, each interface place belongs to a unique portnet.

\begin{definition}[Basic component] A component $B = (N, \Gamma)$ is a basic component if and only if $|I_N| = |I_N| = 1$, $N$ is the closure of an ST-OWN and the following conditions are met:
\begin{itemize}
  \item $\forall x \in I_N \cup O_N, \exists C \in \Gamma : x \in I_C \cup O_C$, i.e. every interface place belongs to a unique portnet;
  \item $\exists C \in \Gamma : i_C = I_N \land f_C = f_N \land \text{ sellside}(B) = \{C\}$, i.e. there exists exactly one sell side portnet, and all other portnets are buy side portnets.
\end{itemize}

Note that a basic component is reducible to the closure of its sell side portnet. This follows directly from Def. 7.

\begin{corollary} Consider a basic component $B = (N, \Gamma)$ and a portnet $C \in \Gamma : \text{ sell}(C)$. Then $N \sim \text{ closure}(C)$.
\end{corollary}

Note that the closure transition allows the basic component to handle more than one service request. Fig. 1 gives an example of a basic component $M = (N, \Gamma)$, where $\Gamma = \{S1, B1, B2\}$ and $S1$ is a sell side portnet. The sell side portnet has two nested buy side portnets: $B1 \subset_X S1$ and $B2 \subset_X S1$. Net $N$ contains a subworkflow net with initial place $q$ and final place $q'$. This subworkflow net can be reduced by nets $B1$ and $B2$. We refer to the resulting net as an orchestration net. Such a net provides the logic behind the order of invocation of the different buy side portnets within a basic component.

Two components are said to be \textit{composable} if and only if the only set of nodes they share are interface places and if this set is not empty, then either they have compatible portnets or they have identical buy side portnets. Note that we require unique sell side portnets.
**Definition 13 (Composable components).** Two components \( X = (N, \Gamma_X) \) and \( Y = (M, \Gamma_M) \) are composable denoted by \( \text{composable}(X,Y) \) if and only if

- \( (P_N \cup I_N \cup O_N \cup T_N) \cap (P_M \cup I_M \cup O_M \cup T_M) = (O_N \cup I_N) \cap (O_M \cup I_M) \), i.e. two components may only share interface places;
- \( \forall C_1 \in \Gamma_N, C_2 \in \Gamma_M: ((O_{C_2} \cap I_{C_1}) \cup (O_{C_1} \cap I_{C_2}) \neq \emptyset \Rightarrow C_1 \triangle C_2) \land ((I_{C_1} \cap I_{C_2}) \cup (O_{C_1} \cap O_{C_2}) \neq \emptyset \Rightarrow C_1 \sim C_2 \land \text{buy}(C_1)) \).

A composition of a set of pairwise composable components is almost a pairwise union of the tuples of this set, except that the interface places belonging to pairs of compatible portnets, now become the internal places of this composition. Furthermore, the set of portnets of this composition is the set of all incompatible portnets. We extend the composition operation to portnets by treating portnets as components. This is possible in the following way: Consider a portnet \( C \), then this portnet is also a component \( (C, \{C\}) \).

**Definition 14 (Composition of components).** The composition of a set \( S \) of pairwise composable components is denoted by \( \text{comp}(S) = (N, \Gamma) \), where \( N = (P_N, I_N, O_N, T_N, i_N, f_N) \) such that

- \( P_N = (\bigcup_{(X,\Gamma') \in S} P_X) \cup (\bigcup_{(X,\Gamma') \in S} I_X \cap \bigcup_{(X,\Gamma') \in S} O_X) \),
- \( I_N = \bigcup_{(X,\Gamma') \in S} I_X \setminus \bigcup_{(X,\Gamma') \in S} O_X \), \( O_N = \bigcup_{(X,\Gamma') \in S} O_X \setminus \bigcup_{(X,\Gamma') \in S} I_X \),
- \( T_N = \bigcup_{(X,\Gamma') \in S} T_X \), \( F_N = \bigcup_{(X,\Gamma') \in S} F_X \),
- \( i_N = \bigcup_{(X,\Gamma') \in S} i_X \), \( f_N = \bigcup_{(X,\Gamma') \in S} f_X \), and
- \( \Gamma = \{ C \in \bigcup_{(X,\Gamma') \in S} \Gamma' \mid \forall C' \in \bigcup_{(X,\Gamma') \in S} \Gamma' : \neg(C \triangle C') \} \).
Corollary 15. The composition of a set of pairwise composable components is again a component.

Note that the composition of one basic component is in fact the basic component itself. Furthermore, \( \text{comp}(S_1 \cup S_2) \neq \text{comp}(S_1 \cup \{\text{comp}(S_2)\}) \), where \( S_1 \) and \( S_2 \) are sets of pairwise composable basic components.

3.2 Architectural Diagram

We now present a graphical notation to represent a composition of components as an architectural diagram of the system. The diagram abstracts away from the underlying control flow and focuses on the relationships between components and the relationships between the portnets of a component. Components are depicted by a rounded rectangle. The portnets of a basic component are represented by a square. All entities are labeled. The dependency relation between a pair of portnets belonging to different components is represented by a directed arrow indicating the direction of communication initiation, i.e. from a buy side portnet to a sell side portnet. A buy side portnet may have at most one outgoing directed edge while a sell side portnet may have zero or more incoming directed edges. The sell side portnet with zero incoming directed edges becomes the portnet of the composition. The portnets of the composition are represented by extending the portnet with a dotted line to the boundary of the composition. By the structure of a component, we know that all the buy side portnets of a basic component are nested within the sell side portnet. Furthermore, a buy side portnet may nest one or more buy side portnets. We represent the nesting of portnets by a dotted directed edge leading from the child to its parent. Note that the Service Component Architecture assembly diagram [3] notation is similar but does not consider nested portnets. We present an example of an architecture diagram in Fig. 2. The figure shows a composition of basic components \( D = \text{comp}(\{A_1, B_1, B_2, B_3, C_1, C_2, C_3\}) \). Basic component \( A_1 \) has a sell side portnet \( P_1 \) that contains three buy side portnets \( P_2, P_3 \) and \( P_4 \). Portnet \( P_4 \) is nested...
in portnet $P_3$. Furthermore, portnet $P_2$ is compatible with portnet $P_5$, $P_3$ with $P_8$ and so on. Note that portnets $P_1$, $P_{15}$ and $P_{17}$ are the portnets of the composition.

4 Behavior

In this section, we study the behavior of a composition of components. In particular, we are interested in weak termination of components, which we define on the skeleton system of the component.

**Definition 16 (Weak termination of a component).** A component $N$ is weakly terminating if its skeleton system weakly terminates.

To prove the weak termination property for an arbitrary composition of components, we need to first show that an arbitrary composition portnets is weakly terminating. In conjunction with the result on weakly termination portnets, we give a sufficient condition and prove that a composition of components satisfying this condition is always weakly terminating.

4.1 Composition of Portnets

In this section, we prove the weak termination property for an arbitrary composition of portnets involving a sell side portnet and a set of one or more pairwise compatible buy side portnets. This means all the buy side portnets are identical copies of each other and the skeletons of each of these portnets are isomorphic to the skeleton of the sell side portnet. Furthermore, for each transition in a buy side portnet there exists an isomorphic partner transition in the sell side portnet with a different direction of communication. Such composition patterns allow us to design a network of communicating components. An example of a composition of pairwise compatible portnets is illustrated in Fig. 3. The sell side portnet specifies the communication protocol for the acceptance of the registration details from compatible buy side portnets. Note that the portnets satisfy both the leg property and the choice property. In Fig. 4, we illustrate two cases of a deadlock in a composition of portnets not satisfying the leg property and the choice property.

We lift the notion of isomorphism to firing sequences in the following way. If two Petri nets are isomorphic with respect to some bijective function $\rho$, then for every executable firing sequence in one of the nets there exists an executable firing sequence in the other net and these firing sequences are isomorphic with respect to the function $\rho$.

We will first prove that for any reachable marking, from the initial marking, in a composition of portnets, there exists a firing sequence in the sell side portnet and a firing sequence in one of the buy side portnet and either these firing sequences are already isomorphic or one of the firing sequence can be extended by firing a sequence of receive transitions and then they are isomorphic.
Furthermore, all other buy side portnets have firing sequences with only send transitions.

In a composition of portnets, from the initial marking, it is always possible that at least one of the buy side portnet fires a sequence of send transitions. The sell side portnet can now fire a sequence of receive transitions enabled by one of the buy side portnet followed by a sequence of send transitions. At this stage, the choice of a buy side portnet is made and the chosen buy side portnet can fire a sequence of receive transitions enabled by the sell side portnet, resulting in a firing sequence that is isomorphic to the firing sequence of the sell side portnet.

Furthermore, the firing sequences of the rest of the buy side portnets contain only send transitions. The crux of the proof lies in the two structural properties of a portnet, namely the leg property and the choice property. The leg property ensures that no portnet has taken a leg one more time than its compatible partner (portnet). The choice property on the other hand ensures that every time one of the portnets makes the choice of a leg, the choice is communicated to the compatible partner. In the absence of these constraints, it is very easy to design a composition of portnets that deadlock.

**Lemma 17.** Consider a composition of portnets $N = \text{comp}(\{A, B_1, \ldots, B_k\})$ with the initial marking $i_N = [i_A, i_{B_1}, \ldots, i_{B_k}]$, where $k \in \mathbb{N}$, $A$ is a sell side portnet and $B_1, \ldots, B_k$ are buy side portnets compatible with $A$. Then for any reachable marking $m \in R(N, i_N)$ and an executable firing sequence $\sigma \in T_N^*: i_N \xrightarrow{\sigma} m$ there exist the firing sequences $\sigma_A \in T_A^*: [i_A] \xrightarrow{\sigma_A} \sigma_j \in T_{B_j}^*: [i_{B_j}] \xrightarrow{\sigma_j}$, where $j \in \{1,\ldots,k\}$ and $\sigma = \sigma_A | \sigma_1 | \ldots | \sigma_k$ such that $\exists l \in \{1,\ldots,k\}$: $\forall j \neq l : \forall t \in \sigma_j : \lambda(t) = \text{send}$ and the one of the following conditions hold

- $\sigma_A \cong \sigma_l$
- $\exists \sigma' \in T_A^* : \sigma_A \cong \sigma_l \circ \sigma'$ where, $\forall t \in \sigma' : \lambda(t) = \text{receive}$
- $\exists \sigma' \in T_A^* : \sigma_l \cong \sigma_A \circ \sigma'$ where, $\forall t \in \sigma' : \lambda(t) = \text{receive}$
Proof. We prove the lemma by induction on the length of $\sigma$. The statements hold for the base case $|\sigma| = 0$ and $m = i_N$. Suppose that it holds for $|\sigma| = n$. Now consider $\sigma'$ with $|\sigma'| = n + 1$ and $\sigma' = \sigma \circ t$ for some $t \in T_N$. So for $\sigma$ the induction hypothesis holds. There are now four cases to consider. See the Fig. 5. Note that for any two firing sequences that are isomorphic with respect to a bijective function $\rho$, we will overload $\rho$ to denote both $\rho$ and $\rho^{-1}$.

Fig. 4. Deadlocked portnets

Fig. 5. The four cases
$t \in T_A \mid k \in \bigcup_{j=1}^{n} T_B$  

| $|\sigma_A| \leq |\sigma_l|$ | Case 1 | $|\sigma_A| > |\sigma_l|$ | Case 3 |
|--------------------------|--------|--------------------------|---------|
|                          |        |                          |        |

**Case 1.** Consider first $|\sigma_A| < |\sigma_l|$. Since $t \in T_A$, let $\sigma_A' = \sigma_A \circ \langle t \rangle$.

By the induction hypothesis, there exists $\sigma'' \in T_A^*$ such that $\sigma_l \cong \sigma_A \circ \langle t' \rangle \circ \sigma''$, where $t' \in T_A$. We need to show that $t' = t$. This means both transitions $t$ and $t'$ are enabled. This is possible only if $\bullet t \cap \bullet t' \neq \emptyset$ since the skeleton satisfies the state machine property. It must be the case that $\lambda(t') = \text{receive}$, since $\sigma_l \cong \sigma_A \circ \langle t' \rangle \circ \sigma''$ and $|\sigma_A| < |\sigma_l|$, so the isomorphic partner transition $\rho(t') \in \sigma_l$ has already fired. Furthermore, since $\bullet t \cap \bullet t' \neq \emptyset$, we have $\bullet \rho(t) \cap \bullet \rho(t') \neq \emptyset$, hence $\lambda(t) = \text{receive}$ by the choice property. Since all buy side portnets are structurally identical copies of each other, there are two cases to consider.

- Transition $\rho(t)$ has fired in some $\sigma_j$, where $l \neq j$. Suppose the first $|\sigma_A|$ transitions of $\sigma_l$ and $\sigma_j$ are not the same but both these subsequences end up marking the place $\rho(\bullet t \cap \bullet t')$. This means $\sigma_l$ and $\sigma_j$ contain different legs. But $\sigma_j$ has only send transitions, so it cannot contain a leg. So one of the firing sequences $\sigma_l$ or $\sigma_j$ are such that one is the prefix of the other. This means the two firing sequences have only send transitions and both $\rho(t)$ and $\rho(t')$ have already fired. But then $\rho(t)$ has also fired in $\sigma_l$. This is our next case.

- Transition $\rho(t)$ has fired in $\sigma_l$. So $\sigma_l$ has fired both $\rho(t)$ and $\rho(t')$ and the interface places of these transitions have one token each. This can only happen if $\sigma_l$ has a loop starting at the place in $\bullet \rho(t) \cap \bullet \rho(t')$ and $\sigma_l$ has taken

---

**Fig. 6.** The Case 1 with $|\sigma_A| < |\sigma_l|$
this loop (i.e. a leg) at least once more than $\sigma_A$. But this cannot be possible since the leg property requires at least two transitions with different communication directions. Therefore $t' = t$.

Consider now $|\sigma_A| = |\sigma_I|$. Since $t \in T_A$, let $\sigma'_A = \sigma_A \circ \langle t \rangle$. It must be the case that $\lambda(t) = \text{send}$ because if $\lambda(t) = \text{receive}$ then it means that transition $\rho(t) \in \bigcup_{j=1}^{k} T_{B_j}$ has fired more times than $t$. We consider two cases.

- Say transition $\rho(t)$ has fired at least one more time in $\sigma_I$. But this is not possible since $|\sigma_I| = |\sigma_A|$ and by induction hypothesis, we have $\sigma_A \cong \sigma_I$. This means all tokens produced by $\sigma_I$ have already been consumed by $\sigma_A$. Therefore $\lambda(t) = \text{send}$ and transition $\rho(t)$ is enabled by the firing sequence $\sigma_I$, i.e. $\rho(t) \cap \sigma^*_I \neq \emptyset$. Hence $\sigma_A \circ \langle t \rangle \cong \sigma_I \circ \langle \rho(t) \rangle$.

- Say transition $t$ has been enabled by some $\rho(t) \in \sigma_{j}$, $j \neq l$ and then one of the following is possible: (a) $\sigma_I$ and $\sigma_j$ contain different legs and they both mark places $\rho(t \cap P_A)$ in their respective portnets. But $\sigma_j$ contains only send transitions, so this is impossible by the leg property. (b) $\sigma_I$ is a prefix of $\sigma_j$. This means both $\sigma_I$ and $\sigma_j$ contain only send transitions. In such a case we may swap the value of $l$ and $j$ and then we are in the case where $|\sigma_I| > |\sigma_A|$.

**Case 2.** We have $t \in \bigcup_{j=1}^{k} T_{B_j}$ and this leads us to two cases.

- Suppose transition $t \in \sigma_{j}$. Let $\sigma'_j = \sigma_I \circ \langle t \rangle$, i.e. $\sigma'_j \cap \mathbb{ullet} t \neq \emptyset$. It must be the case that $\lambda(t) = \text{send}$ because if $\lambda(t) = \text{receive}$ it means $\rho(t) \in T_A$ has fired more times than transition $t$. But this is not possible since $|\sigma_I| \geq |\sigma_A|$ and by the induction hypothesis there exists $\sigma'' \in T_A$ such that $\sigma_I \cong \sigma_A \circ \sigma''$. Therefore $\lambda(t) = \text{send}$. So the isomorphic partner transition $\rho(t) \in T_A$ must be enabled by the firing sequence $\sigma_A \circ \sigma''$. Hence $\sigma_I \circ \langle t \rangle \cong \sigma_A \circ \sigma'' \circ \langle \rho(t) \rangle$.

- Suppose $t \in \sigma_{j}$ and $t \notin \sigma_{I}$. Let $\sigma'_j = \sigma_I \circ \langle t \rangle$, i.e. $\sigma'_j \cap \mathbb{\bullet} t \neq \emptyset$. It must be the case that $\lambda(t) = \text{send}$ because if $\lambda(t) = \text{receive}$ then there exists a transition $\rho(t) \in \sigma_A$ but then $\exists t \in \sigma_I$ that has consumed the interface token. This is a contradiction. Hence $\sigma'_j$ satisfies the property that all transitions have the communication direction send.

**Case 3.** We have $t \in T_A$. Let $\sigma'_A = \sigma_A \circ \langle t \rangle$. By the induction hypothesis $\exists \sigma'' \in T_A^*: \sigma_A \cong \sigma_I \circ \sigma''$. There are now two cases to consider

- Suppose $\lambda(t) = \text{send}$, then there exists $\rho(t) \in T_{B_{j}}$ such that $\lambda(\rho(t)) = \text{receive}$ and $\rho(t)$ can be enabled by $\sigma''$. Therefore $\sigma_A \circ \langle t \rangle \cong \sigma_I \circ \sigma'' \circ \langle \rho(t) \rangle$.

- Suppose $\lambda(t) = \text{receive}$. Since $\sigma_A > \sigma_I$, all tokens produced by transitions in $\sigma_I$ have already been consumed by the transitions in $\sigma_A$, hence $t$ must have been enabled by one of the $\rho(t) \in \sigma_{j}$, where $j \neq l$ and $\lambda(\rho(t)) = \text{send}$. Since $\rho(t)$ has already fired either one of the following is possible: (a) $\sigma_I \circ \sigma''$ and $\sigma_j$ contain different legs but they both mark places $\rho(t \cap P_A)$ in their portnets. But $\sigma_j$ contains only send transitions, so this is impossible by the leg property. (b) $\sigma_I$ is a prefix of $\sigma_j$. This means both $\sigma_I$ and $\sigma_j$ contain only send transitions. In such a case we may swap the values of $l$ and $j$ and then the statement holds.
Case 4. We have $t \in \bigcup_{j=1}^{k} T_{B_j}$. There are two cases to consider.

- Suppose $t \in \sigma_l$. Let $\sigma'_l = \sigma_l \circ (t)$ i.e. $\sigma'_l \cap \bullet t \neq \emptyset$. By the induction hypothesis $\exists \sigma'' : \mathcal{P}_1 : \sigma_A \cong \sigma_l \circ (t') \circ \sigma''$, where $t' \in T_{B_1}$. Since $\sigma'_l = \sigma_l \circ (t)$, we have to show that $t' = t$.

Let $t' \neq t$. This means both $t$ and $t'$ are enabled. By the structure of a portnet (i.e. skeleton is an S-net), this is possible only if $\bullet t \cap \bullet t' \neq \emptyset$. Now $\lambda(t') = \text{receive}$ since $\sigma_A \cong \sigma_l \circ (t') \circ \sigma''$ and $|\sigma_l| \prec |\sigma_A|$ which means $\rho(t') \in \sigma_A$ has already fired. Hence $\lambda(t') = \lambda(t) = \text{receive}$, since $\bullet t \cap \bullet t' \neq \emptyset$. Now $\sigma_A$ is a firing sequence in portnet $A$, it is possible that both $\rho(t)$ and $\rho(t')$ could have fired only if there is a loop starting at $\bullet \rho(t) \cap \bullet \rho(t')$ and $\sigma_A$ has taken this loop (i.e. a leg) at least once more than $\sigma_l$. But this is not possible due to the leg property. Therefore $t' = t$. It follows that $\sigma_A = \sigma_l \circ (t) \circ \sigma''$.

- Suppose $t \in \sigma_j$ and $t \notin \sigma_l$. Let $\sigma'_j = \sigma_j \circ (t)$ i.e. $\sigma'_j \cap \bullet t \neq \emptyset$. By the induction hypothesis $\exists \sigma'' : \mathcal{P}_1 : \sigma_A \cong \sigma_l \circ \sigma''$. Now it must be the case that $\lambda(t) = \text{send}$ because if $\lambda(t) = \text{receive}$ then $\rho(t) \in \sigma_A$ has fired and both transitions $t \in T_{B_1}$ and $t \in T_{B_j}$ must be enabled by $\sigma_l \circ \sigma''$ and $\sigma_j$ i.e., $\bullet t \cap (\sigma_l \circ \sigma'') \circ \sigma_j \neq \emptyset \land (\sigma_l \circ \sigma'') \cap \sigma_j \neq \emptyset$. This means, either $\sigma_l$ is the prefix of $\sigma_j$, in which case we may swap $l$ and $j$ and then the statement holds, or $\sigma_l$ and $\sigma_j$ contain different legs, but this is not possible since $\sigma_j$ has only send transitions. Hence $\lambda(t) = \text{send}$ and is enabled only by firing sequence $\sigma_j$. So $\sigma'_j$ satisfies the property that all transitions have the communication direction send. So the statement holds.

□

Next, we consider the composition of a pair of compatible portnets and show that if the firing sequences of the two portnets are of the same length, then the interface places of this composition are empty.

Lemma 18. Consider a composition of portnets $N = \text{comp}(\{A, B\})$ with initial marking $[i_A, i_B]$, where $A$ and $B$ are two compatible portnets. For a reachable marking $m \in R(N, i_N)$ and an executable firing sequence $\sigma \in T_N : [i_N] \xrightarrow{\sigma} m$, if $|\sigma|_A = |\sigma|_B$ then $m(p) = 0$ for all $p \in I_A \cup O_B$.

Proof. Let $|\sigma|_A = |\sigma|_B$, where $\sigma_A = \sigma|_{T_A}$, $\sigma_B = \sigma|_{T_B}$. From Lemma 17 with $k = 1$, we conclude that $\sigma_A \cong \sigma_B$ and $\forall t \in \sigma$ satisfies $\lambda(t) = \text{send} \Rightarrow \rho(t) \in \sigma : \lambda(\rho(t)) = \text{receive}$. So for every transition that produces a token on an interface place there exists a corresponding transition that consumes it, which leads us to conclude $m(p) = 0$ for all $p \in I_A \cup O_A$, since interface places were empty in the initial marking.

□

The above results in conjunction with the notion of compatibility is used to show that the composition of a pair of compatible portnets, weakly terminates.

Lemma 19. The composition of a pair of compatible portnets weakly terminates.
Proof. Consider a composition of portnets \( N = \text{comp}(\{A, B\}) \), where \( A \) and \( B \) are compatible portnets. Let a reachable marking \( m \in \mathcal{R}(N, [i_N]) \) and a firing sequence \( \sigma : i_N \xrightarrow{\sigma} m \). By Lemma 17, with \( k = 1 \), there exists \( \sigma' : \sigma_A \cong \sigma_B \circ \sigma' \lor \sigma_B \cong \sigma_A \circ \sigma' \) such that \( m \xrightarrow{\sigma'} m' \). Let \( \sigma = \sigma_B \circ \sigma' \) then \( \sigma_A \cong \sigma_B \) and by Lemma 18, \( m'(p) = 0 \) for all \( p \in I_A \cup O_A \), i.e., the interface places are empty. Since portnet \( A \) weakly terminates, there is a firing sequence \( \alpha \) in \( \text{skeleton}(A) \) such that \( m' \xrightarrow{\alpha} [f_A] \). As portnet \( B \) is compatible, \( m'_B \xrightarrow{\rho(\alpha(1))...\rho(\alpha(|\alpha|))} [f_B] \) in \( \text{skeleton}(B) \). We will now build an executable firing sequence \( \beta \) out of \( \alpha \) in the following manner. For all \( i \in \{1...|\alpha|\} \), if \( \lambda(\alpha(i)) = \text{send} \) then \( \beta(2i - 1) = \alpha(i) \land \beta(2i) = \rho(\alpha(i)) \), otherwise \( \beta(2i - 1) = \rho(\alpha(i)) \land \beta(2i) = \alpha(i) \). By Lemma 18, the interface places are empty, so \( m' \xrightarrow{\beta} [f_A, f_B] \). \( \square \)

We will now prove the weak termination property for an arbitrary composition of portnets.

**Theorem 20.** Let \( A, B_1, ..., B_k \) be portnets such that \( \text{sell}(A) \) and \( B_i \triangleq A \) for all \( 1 \leq i \leq k \), then \( \text{comp}(\{\text{closure}(A), B_1, ..., B_k\}) \) weakly terminates.

Proof. Let \( m \in \mathcal{R}(N, [i_A, i_{B_1}, ..., i_{B_k}]) \) and \( \sigma \in T_N^* : [i_A, i_{B_1}, ..., i_{B_k}] \xrightarrow{\sigma} m \). By Lemma 17, \( \exists \sigma \in T_N^* : \sigma_A \cong \sigma \circ \sigma \lor \exists \sigma \in T_A^* : \sigma_i \cong \sigma_A \circ \sigma \), where \( i \in \{1...k\} \), such that \( m \xrightarrow{\sigma} m' \). Let \( \sigma' = \sigma \circ \sigma \). Since portnet \( A \) weakly terminates, from Lemma 19, we may conclude that there exists \( \sigma'' \in T_A^* \) such that \( m' \xrightarrow{\sigma''} [f_A] + [f_i] \). Now \( A \) is the closure of a sell side portnet, so \( \exists t \in T_A \) such that \( [f_A] \xrightarrow{t} [f_A] \). This means \( A \) can handle more than one buy side portnet.

From the remaining firing sequences, we may choose for some \( j \) such that \( j \neq i \), the firing sequence \( \sigma_j \) and this firing sequence contains only send transitions. By Lemma 17, there \( \exists \sigma \in T_A^* : \sigma_j \cong \sigma_A \circ \sigma \) and \( \sigma_A = \epsilon \). Hence, \( \sigma_j \cong \sigma \). Again, by Lemma 19, there is a firing sequence leading to \([f_A] + [f_i] + [f_j]\). So in \( k \) steps \([f_A, f_1, f_2, ..., f_k]\) is reachable. \( \square \)

### 4.2 Composition of Components

We have shown that an arbitrary composition of pairwise compatible portnets always weakly terminates. We build upon these results and show that the same holds for an arbitrary composition of portnets provided the graph of the composition is *acyclic*. This is because the presence of a cycle indicates a deadlock in the composition. An example of a deadlock in a cyclic composition is illustrated in Fig. 7. In this example, basic component \( A \), provides a service that requires the service offered by basic component \( B \). Now basic component \( B \) in turn requires the service offered by \( A \) in order to deliver its own service. This results in a deadlock because component \( A \) is already processing a service.

In our framework, we call an acyclic composition of pairwise composable basic components a *composite component*. Note that we will use the shorthand \( D \) instead of \( D = (N, \Gamma) \), \( D' \) instead of \( D' = (N', \Gamma') \) and so on, to denote a component without explicitly labeling the tuples. We first introduce the notion
Fig. 7. Deadlock in a cyclic composition

of a partner for a buy side portnet in a component, which is the component that provides the compatible sell side portnet.

Definition 21 (Partner). For a non-empty set $S$ of composable basic components, and the set $B$ consisting of all buy-side portnets of the components of $S$, we define the function partner : $S \times B \rightarrow S$ by $\forall D, D' \in S, \forall C \in B : D' = \text{partner}(D, C) \iff \exists C' \in \Gamma' : \text{sell}(C') \land C \equiv C'$.

Definition 22 (Acyclic composition, composite component). Consider a set of $S$ of pairwise composable basic components. Let $R \subseteq S \times S$ be the relation such that $\forall D, D' \in S : (D, D') \in R \Rightarrow \exists C \in \Gamma : D' = \text{partner}(D, C)$. The composition is a composite component if and only if the transitive closure $R^*$ is irreflexive.

We will first show that the composition of a weakly terminating composite component and a basic component such that (a) there exists no buy side portnet in the basic component that is compatible with any sell side portnet of the composite component, (b) there exists at least one buy side portnet in the composite component that is compatible with the sell side port of the basic component and (c) all buy side portnets of the composite component compatible with the sell side portnet of the basic component are subworkflow nets, weakly terminates.

Lemma 23. Consider the set $S$ of pairwise composable basic components such that its composition is a weakly terminating composite component $M = (N, \Gamma)$. Let $Y = (X, \Gamma')$ be a basic component with a sell side portnet $C' \in \Gamma'$, such that there exist no buy side portnet in $Y$ that is compatible with any sell side portnet in $M$, i.e. $\forall \exists C \in \Gamma' : \text{partner}(Y, C) = M$ and there exists at least one buy side portnet in $M$ compatible with the sell side portnet of $Y$, i.e. $\exists C \in \Gamma : \text{partner}(M, C) = Y$. Furthermore, let $\forall C \in \Gamma : (\text{buy}(C) \land C \equiv C') \Rightarrow C \subseteq N$. 18
i.e. all buy side portnets of the composite component are subworkflow nets. Then the composition \( \text{comp}(S \cup \{Y\}) \) weakly terminates.

Proof. Let the sell side port of \( Y \) be the portnet \( S \in I' \) and the set of pairwise compatible portnets of \( M \) be \( B_1, ..., B_k \in I' \). Note that we will use the shorthand \( \sigma_i \) to denote the projection \( \sigma_{iP_{B_i}} \) for some component \( B_i \) and an arbitrary firing sequence \( \sigma \). Furthermore, we will denote by \( \sigma_Z \) the firing sequence \( \sigma_{T_M \setminus (T_{B_1} \cup ... \cup T_{B_k})} \) for a given firing sequence \( \sigma \). Consider the composition \( \text{comp}(S \cup \{Y\}) \). Let \( D = \text{skeleton}(\text{comp}(S \cup \{Y\})) \) and its skeleton system be \( (D, [i_D], [i_D]) \). Consider a reachable marking \( m \in R(D, [i_D]) \) and an executable firing sequence \( \sigma \in T^*_D : [i_D] \xrightarrow{\sigma} m \). We distinguish between two cases.

- **None of the buy side portnets are marked.** This means \( \forall j \in \{1...k\} : \text{occurs}(\sigma, \cdot_{1B_j}) = \text{occurs}(\sigma, \cdot_{f_{B_j}}) \). Now in the composition \( D \), firing sequence \( \sigma \) results in marking \( m \) and this is almost a marking of \( M \). The only difference being that component \( Y \) has one token in its initial or final place. Consider the projection of this marking \( m_M \) on component \( M \). Since \( M \) weakly terminates, there must exist an executable firing sequence \( \tilde{\sigma} \in T^*_M \) such that \( m_M \xrightarrow{\tilde{\sigma}} [f_M] \).

  Without loss of generality, assume \( \forall j \in \{1...k\} : \sigma_j \neq \epsilon \). This means firing sequence \( \tilde{\sigma} \) must contain the interleaving of full traces through each of these buy side portnets, i.e. \( \tilde{\sigma} \in \sigma_1 || ... || \overline{\sigma_T} || \sigma_Z \). We will now transform the firing sequence \( \tilde{\sigma} \) into an executable firing sequence \( \sigma'' \in T^*_D : m \xrightarrow{\sigma''} [f_D] \).

  We reshuffle the transitions of the firing sequences through portnets \( B_1, ..., B_k \) such that the full traces of \( B_1, ..., B_k \) are consecutive subsequences. To verify this, let \( \tilde{\sigma} = \sigma_1 \circ \sigma_a \circ \sigma_2 \circ \sigma_b \circ \sigma_3 \), where \( \sigma_a \) and \( \sigma_b \) are parts of the full trace of some portnet \( B_j \) where \( j \in \{1...k\} \) and no transitions in \( \sigma_1, \sigma_2, \sigma_3 \) belong to \( T_{B_j} \), i.e. \( \forall t \in \sigma_1 \circ \sigma_2 \circ \sigma_3 : t \notin T_{B_j} \). So by the structure of portnets (one entrance, one exit) and the fact that the composition is acyclic, in particular, the fact that no two buy side ports are nested and consume from the same sell side port, we have \( \forall t \in \sigma_a, t' \in \sigma_2 : t'' \cap t' = \emptyset \).

  By the exchange lemma (see [5]), we may exchange \( \sigma_2 \) and \( \sigma_b \) in order to obtain a \( \sigma' = \sigma_1 \circ \sigma_a \circ \sigma_2 \circ \sigma_b \) which is also executable and results in the same marking as \( \tilde{\sigma} \). By repeated application of this transformation, we obtain a firing sequence \( \sigma' \) for the component \( M \) in which all full traces of the portnets \( B_1, ..., B_k \) are consecutive. Note that all traces through a portnet are full traces, since \( \sigma \) ends in the final marking \([f_M] \). It is now possible to transform firing sequence \( \sigma' \) into an executable firing sequence \( \sigma'' \in T^*_D \) of the composition \( D \). We may do so by firing after each send transition of portnet \( B_j \), the mirror transition in \( Y \). This is possible because all portnets have full traces in sequential order and so component \( Y \) can handle them sequentially, as supported by Theorem 20.

- **At least one buy side portnet is marked.** This means \( \exists j \in \{1...k\} : \text{occurs}(\sigma, \cdot_{1B_j}) > \text{occurs}(\sigma, \cdot_{f_{B_j}}) \). We have the case that at least one portnet \( B_j \) for some \( j \in \{1...k\} \) has a marked place. Let \( A \subseteq \{1...k\} \) be the largest
subset such that \( \text{occurs}(\sigma, \bullet i_a) > \text{occurs}(\sigma, \bullet f_{i_a}) \) for all \( a \in A \). Let \( k = |A| \) and note that \( k \geq 1 \).

Consider now the composition of portnets \( Q = \text{comp}(\{ B_1 \} \cup \ldots \cup \{ B_k \} \cup \{ S \}) \). Now \( Q \) is a subnet of component \( D \). Consider the marking \( m_Q \), i.e. the projection of marking \( m \) on the subnet \( Q \). Let a firing sequence \( \sigma_Q \in T_Q^* \) such that \( \sigma_Q \stackrel{\alpha}{\rightarrow} m_Q \). By Lemma 17, there is one buy side portnet \( B_1 \), such that \( \exists \sigma' \in T_B^* : \sigma_S \equiv \sigma_1 \circ \sigma' \) or \( \exists \sigma' \in T_S^* : \sigma_1 \equiv \sigma_S \circ \sigma' \).

Now we may extend firing sequence \( \sigma \) by \( \sigma' \), i.e. \( \sigma \circ \sigma' \) because the transitions of portnet \( B_1 \) do not share interface places with the other portnets \( B_2, \ldots, B_k \).

So we obtain a firing sequence for \( D \) with subsequences in portnets \( B_1 \) and \( S \) and they are of the same length. Now we may continue to the final marking of \( B_1 \), by firing the mirrored transitions in the right order (i.e. send transitions before a receive transition). In this manner, we derive a firing sequence \( \sigma'' \in T_{B_1}^* \cup \ldots \cup T_{B_k}^* \cup T_{S}^* \) such that \( \sigma \circ \sigma' \circ \sigma'' \) and this contains the full trace through portnet \( B_1 \). It is now possible to fire the closure transition of \( S \), so portnet \( S \) is again in its initial marking. In fact, we reduced the first element from set \( A \). We may repeat this procedure for all portnets \( B_2, \ldots, B_k \) and we end up with an executable firing sequence \( \sigma' \in T_{D}^* \) and this does not mark any of the buy side portnets anymore. Now we are in the previous case and we may apply the construction described there to reach the final marking.

Hence skeleton system of the composition \( D \) weakly terminates. \( \square \)

We will now show that an arbitrary composite component weakly terminates.

**Theorem 24.** A composite component weakly terminates.

**Proof.** Consider the set \( S \) of pairwise composable basic components such that \( \text{comp}(S) \) a composite component. Let \( R \subseteq S \times S \), such that \( \forall D, D' \in S : (D, D') \in R \Rightarrow \exists C \in \Gamma : D' = \text{partner}(D, C) \).

Let \( Q \) be the set of all portnets of all components that belong to the set \( S \). We define a function \( \text{sellside} : S \rightarrow Q \) such that \( \forall D \in S, C \in \Gamma : C = \text{sellside}(D) \leftrightarrow \text{sell}(C) \). Note that the graph of the composition of basic components is a directed acyclic graph. So we define a topological sort \( \preceq \) over the set \( S \) such that for all \( D, D' \in S \), we have \( D \preceq D' \) if and only if:

\[
\neg (D', D) \in R^* \land \forall D'' \in S, C_1, C_2 \in \Gamma : C_1 \triangleleft_D C_2 \land C_1 \triangleleft \text{sellside}(D'') \land C_2 \triangleleft \text{sellside}(D') : D' \preceq D''
\]

Let \( S = \{ S_1, S_2, \ldots, S_n \} \) such that \( \forall 1 \leq i \leq j \leq n : S_i \preceq S_j \). We will prove the theorem by structural induction. The base case holds, since \( S_1 \) is a basic component. By [9] the composition \( \text{comp}(S_1) \), weakly terminates, because a basic component is the closure of a ST-OWN.

Assume that for some \( i < n \) the composite component \( N = \text{comp}(\{ S_j \mid 1 \leq j \leq i \}) \), weakly terminates. Now consider the composition \( N' = \text{comp}(\{ S_j \mid 1 \leq j \leq i + 1 \}) \). Note that the relation \( \preceq \) introduces an order in the composition of basic components. So it is possible to apply the reduce operation (Def. 4),
on the composition \( N' \), until all buy side portnets that are compatible with the \( \text{sellside}(S_{j+1}) \) are subworkflow nets of the composition. This results in a composite component \( N'' \). By Lemma 23, the reduced component \( N'' \) weakly terminates and from Lemma 8, we can conclude that \( N' \) also weakly terminates.

\[ \square \]

5 Construction Method

This section presents a construction method that derives a composition of basic components from an architectural diagram and ensures that the derived composition is weakly terminating. The construction method is based on place refinement and composition as defined in the previous sections.

The construction method starts with an architecture diagram of a composite component, like the one depicted in Fig. 2. To construct a basic component we require the three ingredients, namely a sell side portnet, a set of buy side portnets and a set of orchestration nets (ST-WFN). An orchestration net is used to elaborate the activities of a basic component by being able to introduce internal activities, concurrency and choice in a structured way. Furthermore, the places introduced by an orchestration net, may be refined with buy side portnets during construction, thereby allowing us to model both the choice of service invocations and concurrency in service invocations.

First, for each basic component in the diagram, design the sell side portnet. Next, for each basic component in the diagram, derive all its buy side portnets from existing compatible sell side portnets. Note that a buy side portnet may be derived from a sell side portnet by changing the direction of communication associated with each transition in the corresponding sell side portnet. Lastly, for each basic component in the diagram, design the necessary orchestration nets that will be required during the construction.

We may now convert all the sell side portnets into a basic component by introducing the closure transition. For each basic component, the architecture diagram gives the order of nesting of its portnets. Using this information, we may now start designing the control flow of a basic component by successive refinements of an existing internal place with either an orchestration net or a buy side portnet, until all the buy side portnets of the basic component have been added in the right order of nesting and the desired basic component has been constructed.

Construction method

1. Design an architecture diagram for an acyclic composition of basic components using the techniques of Sec. 3.
2. Design all the portnets and orchestration nets that we will need for this composition.
3. For each node in this architecture diagram select the corresponding sell side portnet and apply the closure operation.
4. For each basic component, repeat the following steps until in each basic component all the buy side portnets have been added in the right order of nesting, and the desired orchestration has been constructed:
   (a) If an orchestration needs to be added first, then choose an internal place and refine with the right orchestration net;
   (b) Otherwise, choose an internal place and refine with a buy side portnet in the order defined by the architectural diagram;
5. Compose the set of basic component using the composition operation.

**Theorem 25.** The construction method always results in a composite component that weakly terminates.

**Proof.** Let $orch$ denote the refinement of a place by an orchestration net, $port$ denote the refinement of a place by a portnet, $close$ denote the closure operation over a set of sell side ports and $comp$ denote the composition operation. For clarity, we denote this set of refinements by the alphabet $\Sigma = \{close, orch, port, comp\}$. The construction method is in fact a sequence of refinements expressed by the language $L = close; L'; comp$; where $L' = (orch^*; port)^*; orch^*$; and results in a composition of basic components. Consider a feasible sequence of refinements $\sigma \in L$ where $\sigma = close; \sigma'; comp$. The successive elimination of the tail element of the subsequence $\sigma'$ can be seen as analogous to successive applications of the reduce operation (see Def. 3). By Lemma 8, we know such a reduction preserves weak termination. We repeat this until we have completely reduced this composition and so it weakly terminates. $\square$
5.1 Construction of Orchestration Nets and Portnets

For the construction of portnets, we extend the Jackson refinement rules $R_0$, $R_1$, $R_2$, and $R_3$ with interface places as depicted in Fig. 8. Note that rule $R_0$ is a special case of the refinement rule of Def. 3. Rule $R_0'$ and $R_0''$ extend rule $R_0$ such that the refining transition can have either the communication direction send or receive. The extensions of rule $R_3$ maintains the leg property by only adding loops with different directions of communication. Similarly, Rule $R_2$, which adds a choice to the net is extended such that the choice property is maintained. Rule $R_1$ is extended to allow two way communication.

The construction of an orchestration net starts with a single place. By applying the refinement rules of [8], we obtain larger nets that are guaranteed to be weakly terminating. We limit ourselves by applying the Jackson refinement rules $R_0$, $R_1$, $R_2$, $R_3$, $R_4$ such that the result remains an ST-net. The construction of a portnet starts with the choice of the sell side portnet or buy side portnet. A sell side portnet is obtained by the sequence of refinements: $R_0; R_1; R_1'$. A buy side portnet is obtained by the sequence of refinements: $R_0; R_1; R_1''$. We may now further elaborate these portnets by arbitrary applications of the refinement rules $R_0; R_0'$, $R_0; R_0''$, $R_1; R_1'$, $R_1; R_1''$, $R_2; R_2'$, $R_2; R_2''$, $R_3; R_3'$, $R_3; R_3''$, while ensuring that the structure of the portnet remains a S-OWN by not allowing for place duplication (rule $R_4$). Note that the choice of the place to apply the refinement sequence $R_3; R_3'$ or $R_3; R_3''$ must be such that the newly introduced legs do not violate the leg property.

Theorem 26. The refinement rules for portnets preserve the properties of a portnet.

Proof. The refinement rules for portnets excludes place duplication. So the skeleton of the refined net is still a S-net. Furthermore, the leg property and choice property are preserved by construction. \qed

6 The Control Flow of an Autonomous Mobile Robot

In this section, we apply the construction method to design the control flow of the navigation system on a mobile robot. The system comprises of the four main components: the user interface, the navigation system, the platform controller and the laser scan controller. The robot perceives its environment by means of a planar laser scanner. The laser scan controller provides the latest scan as a service. The platform controller is a composite component and provides two services (a) to set a desired velocity (b) queries on the latest odometry. The navigation system is capable of creating a map of its environment and localizing itself on this map using the current laser scan and odometry services. Furthermore, the navigation system can accept a waypoint and generate a sequence of velocity commands that drive the platform to this waypoint while avoiding obstacles. The user interface at the remote location allows an operator to visualize this map and give waypoint to the navigation system. While a waypoint is in
progress an operator receives feedback on the progress of this goal. Once the robot has reached its waypoint, the operator is notified. An architecture of the system is presented in Fig. 9.

The navigation system is a composite component comprising of the navigation manager and planning. The latter is again a composite component and comprises of the global planner, the local planner and the mapping and localization system. The mapping and localization system is capable of generating a map and localizing itself using the services offered by the laser scan controller and platform controller. The global planner accepts a map and a waypoint goal and generates a global plan (trajectory) from the robot’s current location to the waypoint goal. The local planner accepts a map and a global plan and generates a collision free sequence of velocity commands that drive the robot to the desired waypoint goal. The local planner generates these sequence of velocity commands in a loop until the destination is reached or a valid plan could not be found. In each cycle, the local planner makes use of the mapping and localization system to check its current location and generates feedback on the progress of this goal. If the destination has arrived then this is notified and the planner terminates. If at any moment, a valid velocity command could not be found then this situation is notified and the planner terminates. The navigation manager provides the waypoint navigation as a service to the user interface by orchestrating the components of the navigation system in the right order. The Fig. 10 presents five portents and one orchestration net. From the architecture diagram in Fig. 9, we know portnet $P_5$ is nested in $P_{12}$ and all buy side portnets are nested in the sell side portnet $P_4$. We may now apply the construction method to derive the navigation manager in the following way: $((\text{closure}(P_4) \circ_{r_2} O_1) \circ_{r_4} P_{11}) \circ_{r_5} P_{13}) \circ_{r_1} (P_{12} \circ_{r_3} P_5)$. 

![Fig. 9. Architectural diagram](image-url)
7 Conclusions

In this paper, we introduced a compositional component framework and a construction method to design the control flow of a network of components, while guaranteeing weak termination. The two main concepts of this framework are portnet and basic component. A portnet models the interface of a basic component as a state machine which describes the communication protocol underlying a service negotiation. A basic component provides a service by orchestrating its portnets in the right way. The weak termination property was then investigated by first considering compositions of portnets. It turns out that any pair of compatible portnets that satisfy the leg property and the choice property always weakly terminate. Furthermore, we prove that an acyclic composition of basic components also known as a composite component weakly terminates.

In [6, 12], the authors focus on constructing deadlock free systems using labeled transition systems, i.e., each component is a state machine, which after composition guarantee deadlock freedom. On the other hand, Petri nets offer a natural way to make formal models of the control flow of a software system. The Petri net based construction method provides a structured way to design these control flows and guarantee weak termination by construction. In this way they can focus more on the design of each component without having to worry about deadlocks that could be introduced by a composition of components. The design-
ers of software systems can use the guiding principles defined by the construction method during system design.

As future work, we intend to extend this framework to model the software components of control systems. Furthermore, as time is a critical factor in the safety of any control system, we intend to extend the construction procedure to guarantee weak termination within time constraints.

References


