Branching processes of communication structured PT-nets

Bowen Li
Branching processes of communication structured PT-nets

B. Li

Abstract

Communication structured occurrence nets (CSON) are an extension of occurrence nets. They can be used to represent the execution behaviours of complex evolving systems. Communication structured place transition nets (CSPT-nets) provide a system-level model for describing the interaction between different systems, and CSONs can model individual runs of CSPT-nets.

In this paper, we investigate branching processes of CSPT-nets which provide a complete information about their behaviours. We also outline an algorithm for the construction of unfoldings of CSPT-nets.
Bibliographical details

LI, B.
Branching processes of communication structured PT-nets
[By] B. Li
(Newcastle University, Computing Science, Technical Report Series, No. CS-TR-1389)

Added entries

NEWCASTLE UNIVERSITY

Abstract

Communication structured occurrence nets (CSON) are an extension of occurrence nets. They can be used to represent the execution behaviours of complex evolving systems. Communication structured place transition nets (CSPT-nets) provide a system-level model for describing the interaction between different systems, and CSONs can model individual runs of CSPT-nets.

In this paper, we investigate branching processes of CSPT-nets which provide a complete information about their behaviours. We also outline an algorithm for the construction of unfoldings of CSPT-nets.

About the authors

Bowen Li is a PhD student and part-time research assistant in Newcastle University, School of Computing Science.

He is working on the EPSRC funded project UNCOVER (UNderstanding COmplex system eVolution through structurEd behaviours). An overall goal of UNCOVER is to develop a rigorous methodology supported by a toolkit based on structured occurrence nets, in order to provide an effective approach to acquiring and exploiting behavioural knowledge of a complex evolving system.

Suggested keywords

BRANCHING PROCESSES
OCCURRENCE NETS
STRUCTURED OCCURRENCE NETS
COMMUNICATION STRUCTURED PT-NETS
Branching processes of communication structured PT-nets

Bowen Li
School of Computing Science, Newcastle University
Newcastle upon Tyne NE1 7RU, UK
Email: bowen.li@ncl.ac.uk

Abstract—Communication structured occurrence nets (CSON) are an extension of occurrence nets. They can be used to represent the execution behaviours of complex evolving systems. Communication structured place transition nets (CSPT-nets) provide a system-level model for describing the interaction between different systems, and CSONs can model individual runs of CSPT-nets. In this paper, we investigate branching processes of CSPT-nets which provide a complete information about their behaviours. We also outline an algorithm for the construction of unfoldings of CSPT-nets.

I. INTRODUCTION

Complex evolving systems consist of a large number of sub-systems which may proceed concurrently and interact with each other. Such systems generally suffer from high complexity of behaviours of not only each single sub-system but also the dependencies between them. The communication between sub-systems may either be synchronous or asynchronous. Usually, the former implies that a sender waits for an acknowledgement of a message before proceeding, while in the latter the sender proceeds without waiting. The standard Petri net approach represents asynchronous relation, but does not provide means to directly synchronise different transitions.

Communication structured occurrence nets (CSON) [1], [2] are a Petri-net based formalism that can be used to model executions of complex evolving systems. The model has the capability of providing a meaning of the synchronous interaction between communicated systems. The concept of CSON extends that of occurrence nets [3] which are used to represent execution behaviours of concurrent systems. More precisely, a CSON model combines multiple related occurrence nets into a single structure by letting them communicate via two special relationships, viz., synchronous and asynchronous communications. Just as occurrence net model, CSON essentially is a behaviour-level class of Petri nets, which records the concurrent behaviours of complex evolving systems, and in which underlying structure is acyclic and conflict-free. Communication structured place transition nets (or CSPT-nets), introduced in [4], capture a system-level counterpart of CSONs. CSPT-net is an extension of place transition nets (PT-nets), which uses channel places to combine different PT-nets. A CSON models a single execution of a CSPT-net.

In PT-net theory, branching processes are used to capture complete information of system execution [5], [6]. Intuitively, branching processes act as a ‘bridge’ between net systems and their processes. In communication structured net theory, there is no corresponding ‘bridge’ w.r.t. the branching processes of CSPT-nets. In this paper, we will fill the gap between CSPT-nets and CSONs by defining a new kind of branching processes. Moreover, we outline an algorithm for constructing unfoldings of CSPT-nets.

This paper is organised as follows. Section 2 contains basic notions, in particular, the definitions of communication structured net theory. Branching processes of CSPT-nets are introduced in Section 3. In Section 4, we outline an algorithm for CSPT-net unfolding. Section 5 discusses future work and concludes the paper.

II. BASIC DEFINITIONS

We assume that the reader is familiar with Petri nets and their unfoldings, which can be found in, e.g., [3], [5], [6].

A. PT-nets

A net is a triple \((P, T, F)\) such that \(P\) and \(T\) are disjoint sets of respectively places and transitions (collectively referred to as nodes), and \(F \subseteq (P \times T) \cup (T \times P)\) is the flow relation. The inputs and outputs of a node \(x\) are defined as \(\cdot x = \{ y \mid (y, x) \in F \}\) and \(x^* = \{ y \mid (x, y) \in F \}\). It is assumed that the inputs and outputs of a transition are nonempty sets.

A place transition net (or PT-net) is a tuple

\[
PT = (P, T, F, M_0)
\]

such that \((P, T, F)\) is a net and \(M_0 : P \rightarrow \mathbb{N}\) is an initial marking (in general, a marking is a multiset of places). A step \(U\) is a multiset of transitions. It is enabled at a marking \(M\) if \(M(p) \geq \sum_{t \in P} U(t)\), for every place \(p\). In such a case, the execution of \(U\) leads to the marking \(M'\) given by \(M'(p) = M(p) - \sum_{t \in P} U(t) + \sum_{t \in P} U(t)\), for every \(p \in P\). The notion of a reachable marking is then defined as usual.

B. CSPT-nets

We now introduce an extension of PT-nets which combines several nets into one model by using channel places. Generalising the definition from [4], a communication structured place transition net (or CSPT-net) is a tuple

\[
CSPT = (PT_1, \ldots, PT_k, Q, W, M_0) \quad (k \geq 1)
\]

where each \(PT_i = (P_i, T_i, F_i, M_i)\) is a PT-net, \(Q\) is a set of channel places, \(M_0\) is a set of channel places with initial
marking and \( W \subseteq (T \times Q) \cup (Q \times T) \) where \( T = \bigcup_{i \geq 1} T_i \). It is assumed that the \( PT_i \)'s are disjoint and, for every channel place \( c \), the following are satisfied:

- the nonempty sets of input transitions \( *c \) and output transitions \( c^* \) belong to two distinct component PT-nets;
- if \( *c \subseteq T_i \) (resp. \( c^* \subseteq T_i \)) then there is no reachable marking in \( PT_i \) which enables a step comprising two transitions in \( *c \) (resp. \( c^* \subseteq T_i \)) or two copies of the same transition in \( *c \) (resp. \( c^* \subseteq T_i \)).

The above means that the occurrences of transitions in \( PT_i \) such that \( *c \subseteq T_i \) (resp. \( c^* \subseteq T_i \)) are totally ordered in any execution of \( PT_i \). In other words, we assume that both the output access and input access to the channel places is sequential.

The initial marking \( M_{\text{init}} \) of \( CSPT \) is the sum of all the \( M_i \)'s, including \( M_0 \), and a marking in general is a multiset of places. The execution semantics is defined as before except that a step of transitions \( U \) is enabled at a marking \( M \) if \( M(p) \geq \sum_{t \in \mathcal{P}} U(t) \), for every non-channel place \( p \), and

\[
M(c) + \sum_{t \in *c} U(t) \geq \sum_{t \in c^*} U(t),
\]

for every channel place \( c \). The second condition for step enabledness caters for synchronous behaviour as step \( U \) can count not only on tokens that are already available in channel places at marking \( M \), but also can use the tokens deposited there by other transitions from \( U \) during the execution of \( U \). In this way transitions from \( U \) can help each other and synchronously pass resources to each other.

Figure 1(a) shows a CSPT-net with two component PT-nets and three channel places. The execution between transitions \( t_1 \) and \( t_3 \) can be either asynchronous (\( t_1 \) occurs before \( t_3 \)), or synchronous (\( t_1 \) and \( t_3 \) occur simultaneously). It is interesting to observe that transitions \( t_2 \) and \( t_5 \) are connected by a pair of empty channel places, \( q_2 \) and \( q_3 \), forming a cycle. This indicates that these two transitions can only be executed synchronously. They will be filled and emptied synchronously when both \( t_2 \) and \( t_5 \) participate in one enabled step.

### C. Branching processes of PT-nets

Let \( (P, T, F) \) be a net. Then \( x, x' \in P \cup T \) are in conflict, denoted by \( x \# x' \), if there exist distinct transitions \( t, t' \in T \) such that \( *t \cap *t' \neq \emptyset \) and \( (t, x) \in F^+ \) and \( (t', x') \in F^+ \). A node \( x \) is in self-conflict if \( x \# x \).

A net \( ON = (P, T, F) \) is a branching occurrence net if \( F \) is acyclic; \( |p| \leq 1 \) for all \( p \in P \); no \( t \in T \) is in self-conflict (\( P \) and \( T \) are also called conditions and events respectively); and for every node \( x \), there are finitely many \( y \) such that \( (y, x) \in F^+ \). The set of all places with no inputs \((|p| = 0) \) is the default initial state, denoted by \( \text{Min}(ON) \).

In general, a state is any set of places. If, in addition, \( |p^*| \leq 1 \) for all \( p \in P \), then \( ON \) is a non-branching occurrence net. In branching occurrence net, once we have two branches going out of a place or transition, they will never meet again neither by coming to the same place (the pre-sets of places are at most singleton sets) nor by coming to the same transition (transitions cannot be in self-conflict).

A branching process of a PT-net \( PT \) as in (1) is a pair \( \Pi = (ON, h) \) such that \( ON = (P', T', F') \) is a branching occurrence net and \( h : P' \cup T' \rightarrow P \cup T \) is a mapping such that:
- \( h(P') \subseteq P \) and \( h(T') \subseteq T \);
- for all \( t \in T' \), the restriction of \( h \) to \( *t \) is a bijection between \( *t \) and \( h(t) \), and similarly for \( *t \) and \( h(t)^* \);
- the restriction of \( h \) to \( \text{Min}(ON) \) is a bijection between \( \text{Min}(ON) \) and \( M_0 \);
- for all \( t, v \in T' \), if \( *t = v \) and \( h(t) = h(v) \) then \( t = v \).

The branching process of system \( b \) in Figure 1(a) is given in Figure 2. A maximal branching process of \( PT \) always exists and is called the unfolding of \( PT \) [5].

### D. Non-branching processes of CSPT-nets

Let \( CSPT \) be a CSPT-net as in (2). A non-branching process of \( CSPT \) is a tuple:

\[
\text{CSON} = ((ON_1, h_1), \ldots, (ON_k, h_k), Q', W', h') \quad (k \geq 1)
\]

such that each \( ON_i = (P_i', T_i', F_i') \) is a non-branching occurrence net and \( (ON_i, h_i) \) is a process of \( PT_i \); \( Q' \) is a set of channel places; \( W' \subseteq (T' \times Q') \cup (Q' \times T') \) where \( T' = \bigcup_{i \geq 1} T_i' \); and \( h' : Q' \rightarrow Q \). Moreover, it is assumed that the following are satisfied (below \( h \) denotes the union of \( h_i \) and the \( h_i \)'s, and \( F' = \bigcup_{i \geq 1} F_i' \)):

- \( ON_1, \ldots, ON_k \) are mutually disjoint;
- for every \( q \in Q', |q_i| \leq 1 \) and \( |q'| \leq 1 \);
- the relation \((\subseteq \cup \prec \subset)\prec \circ (\subseteq \cup \subset)\) is irreflexive, where \( \prec \) if there is \( p \in \bigcup_{i \geq 1} P_i' \) with \( p \in t^* \cap *v \), and \( \subset v \) if there is \( q \in Q' \) with \( p \in t^* \cap *v \);
- \( h(\text{Min}(\text{CSON})) = M_{\text{init}} \); where the default initial state \( \text{Min}(\text{CSON}) \) is the sum of initial states in all occurrence nets together with the channel places without inputs.

The above definition extends that in [4] by allowing an infinite number of nodes and provides a meaning of a single run of CSPT-net. As an example, the CSON in Figure 1(b) shows...
one of the processes of the CSPT-net shown in Figure 1(a) when transition $t_3$ is chosen to execute. Again, the interaction between events $e_1$ and $e_3$ are a/synchronous, while $e_2$ and $e_4$ execute synchronous communication.

III. BRANCHING PROCESSES OF CSPT-NETS

In this section, we introduce branching processes for CSPT-nets containing complete reachability related information. The main idea is to combine independent branching processes of the PT-nets contained in a CSPT-net using channel places. In doing so, we also apply a technique derived from merged processes [7], in order to reduce the complexity of the structure.

Let $CSPT$ be a CSPT-net as in (2). Moreover, in order to simplify the presentation, we assume that $M_0$ is empty. A branching process of CSPT is a tuple

$$\Pi_{CSPT} = (\Pi_1 \ldots \Pi_k, Q', W', h)$$

where each $\Pi_i = ((P'_i, T'_i, F'_i), h_i)$ is a branching process of $PT_i$; $Q'$ is a set of channel places; $W' \subseteq (T' \times Q') \cup (Q' \times T')$ where $T' = \bigcup_{i \geq 1} T'_i$; and $h' : Q' \rightarrow Q$. Moreover, it is assumed that the following are satisfied (below $h$ denotes the union of $h'$ and the $h_i$'s):

- $\Pi_1, \ldots, \Pi_k$ are mutually disjoint;
- each $q' \in Q$ has at least one input, and $q' \cup q$ comprises all $t \in T'$ such that $h(t) \in \ast h(q') \cup h(q')$ and they have the same number of $v \in T'$ labeled by transitions in $\ast h(q') \cup h(q')$ satisfying $(v, t) \in F'$;
- for all $t \in T'$, the restriction of $h$ to $t$ is a bijection between $t'$ and $\ast h(t)$, and similarly for $t'$ and $h(t)$;
- $\Pi_{CSPT}$ is covered by a set of non-branching processes of CSPT (in graph-theoretic sense).

The default initial state $Min(CSON)$ is the sum of initial states in all branching processes. There is a unique maximal branching process $\Pi_{CSPT}^{max}$, called the unfolding.

Intuitively, in the above definition we require that the label of every input and output event of a channel place in $\Pi_{CSPT}$ matches corresponding transition in CSPT-net, as well as the occurrence depths of events inserting tokens to a channel place are the same, and are equal to the occurrence depths of events removing the tokens. The occurrence depth of an event $e \in T'$ is the maximum number of events which have same labeled input/output channel place on any directed path starting at initial state and terminating at $e$. We implicitly assume that tokens flowing through channel places are removed according to a FIFO policy. In this way, the size of the representation of the behaviours of a CSPT-nets is kept low.

The $\Pi_{CSPT}$ of Figure 1(a) is given in Figure 2, where $h$ provides the labels of nodes. For ease of view, the dashed lines indicate the flow relation $W'$. It can be seen that each independent CSPT-net preserves the structures of its own unfolding and the relations of all channel place that connect branching processes are preserved as well. The occurrence depths of $e_2$ is $d(e_2) = 2$ since it is the second input event of $q_1$ labeled channel place on the path starting from initial state $p_1$. Note that although $q_1$ has only one output event in the CSPT-net, there are two output events $e_2$ and $e_10$ of channel place $a$ in corresponding CSPT branching process. This is because both $e_2$ and $e_10$ have same labels $t_3$, and also, their occurrence depths are the same, and are equal to the input event of $a$. In the other words, $(e_1, e_2)$ and $(e_7, e_10)$ belong to the first a/synchronous communication between two systems in the CSPT-net. On the other hand, there is no connection between $(e_7, e_10)$ since $e_7$ is an event occurred in the second execution of system $a$ ($d(e_7) = 2$), while $e_10$ occurs in the first execution ($d(e_10) = 1$).

IV. AN ALGORITHM FOR UNFOLDING CSPT-NETS

We now outline an algorithm for constructing branching process of CSPT-nets (see Figure 3). For brevity, in the input CSPT-net it is assumed that each channel place has one input transition and one output transition. The algorithm starts with $\Pi_{CSPT}$ having the channel places and conditions corresponding to the initial markings in CSPT-net. Then the main idea is based on the basic unfolding algorithm such that each possible extended event (defined similarly as in [6]) $e = (t, B)$ is added to $\Pi_{CSPT}$ together with their output conditions. In the case that the corresponding transition of $e$ in CSPT-net is the input of a channel place $q$, then: if there exist a channel place in $\Pi_{CSPT}$ which has the same label with $q$ and occurrence depth as $e$, we create a connection between them. Otherwise a new instance of channel place $q$ is appended to $\Pi_{CSPT}$ with the arcs between new channel place and all existing events which, again, have the same labels and occurrence depths. Similarly, for the new extended event which is the output transition of a channel place in CSPT-net, a connection from matched and existing channel place to the event is created. If there is no such a channel place, we simply move to the next possible extension.
Require: A CSPT-net $CSPT = (PT_1, ..., PT_k, Q, W, M_0)$, where $(k \geq 1)$

Ensure: The unfolding $Unf$ of $CSPT$

$Unf :=$ the empty unfolding

add instances of the places from $M_{init}$ to $Unf$

pe $\leftarrow PE(Unf)$

begin:

while pe $\neq 0$ do

append to $Unf$ an event $e = (t, B)$ of pe

if $h(e)^* \cap Q \neq \emptyset$ then

for all $q \in h(e)^* \cap Q$ in $CSPT$ do

OutputChannelPlace()

if $h(e)^* \cap Q = \emptyset$ then

for all $q \in h(e)^* \cap Q$ in $CSPT$ do

InputChannelPlace()

add new instances of the places from $h(e)^*$ to $Unf$

pe $\leftarrow PE(Unf)$

function OutputChannelPlace()

for all channel places $q' \in Q'$ in $Unf$ do

if $h(q') = q$ and $d(e) = d(q'^*)$ then

add a new arc $(e, q')$

return

for all events $e' \in E$ in $Unf$ do

if $h(q_{new}) \neq \emptyset$ and $d(e') = d(q_{new})$ then

add a new arc $(q_{new}, e')$

return

function InputChannelPlace()

for all channel places $q' \in Q'$ in $Unf$ do

if $h(q') = q$ and $d(e) = d(q'^*)$ then

add a new arc $(q', e)$

return

Figure 3. CSPT unfolding algorithm.

V. CONCLUSION

We defined branching processes for CSPT-nets — a representation of a CSPT-net’s behaviour containing a complete informative w.r.t. reachable markings. Branching processes compose independent branching processes via channel places, where the input and output events of a channel place depend on their labels and occurrence depths. We then outlined an algorithm for constructing the maximal branching process (unfolding).

In future we first plan to investigate the case of a CSPT-net containing channel places with initial markings. Considering the branching process in Figure 4, the channel place $q_1$ has an initial token which must be consumed by $t_3$ firstly. The communication between the two sub-systems then occurs when the occurrence depth of $e_5$ is higher than $e_1$, i.e. $d(e_1) + 1 = d(e_5)$. We also intend to investigate finite complete prefixes of branching processes of CSPT-net. For example, Figure 4(b) is a finite complete prefix of the CSPT-net in Figure 4(b). Finally, we plan to implement a verification method in an existing tool platform [8].

ACKNOWLEDGMENT

I would like to thank my supervisor Maciej Koutny for his comments and help and Victor Khomenko for suggesting the use of merged places in the branching processes of CSPT-nets.

REFERENCES


