Abstract. Petri nets are a well-known model of concurrency and provide an ideal setting for the study of fundamental aspects in concurrent systems. Despite their simplicity, they still lack a satisfactory causally reversible semantics. We develop such semantics for Place/Transitions Petri nets (P/T nets) based on two observations. Firstly, a net that explicitly expresses causality and conflict among events, for example an occurrence net, can be straightforwardly reversed by adding a reverse transition for each of its forward transitions. Secondly, given a P/T net the standard unfolding construction associates with it an occurrence net that preserves all of its computation. Consequently, the reversible semantics of a P/T net can be obtained as the reversible semantics of its unfolding. We show that such reversible behaviour can be expressed as a finite net whose tokens are coloured by causal histories. Colours in our encoding resemble the causal memories that are typical in reversible process calculi.

1. Introduction

Reversible computing is attracting interest for its applications in many fields including hardware design and quantum computing [VBR18], the modelling of bio-chemical reactions [CL11, PUY13, KU16, Pin17, KU18, KAC+20], parallel discrete event simulation [SOJJ18] and program reversing for debugging [GLM14, LNPV18, HUY18, HLN+20].

A model for reversible computation features two computation flows: the standard forward direction and the reverse direction, which allows us to go back to any past state of the computation. Reversibility is well understood in a sequential setting in which executions...
are totally ordered sets of events (see [Lee86]): a sequential computation can be reversed by successively undoing the last not yet undone event. Reversibility becomes more challenging in a concurrent setting because there is no natural way to define a total ordering on concurrent events. Many concurrency models represent the causal dependencies among concurrent events using partial orders. Reversing an execution of a partially ordered set of events then reduces to successively undoing one of the maximal events which has not been undone yet. This is the basis of the causal-consistent reversibility [DK04, PU07b, LMT14, LPU20], which relates reversibility with concurrency and causality. Intuitively, this notion stipulates that any event can be undone provided that all its consequences, if any, are undone beforehand. Reversibility in distributed systems, for example in checkpoint/rollback protocols [VS18] and in transactions [DK05, LLM+13, MMPY20], can also be modelled by causal-consistent reversibility. The interplay between reversibility and concurrency and causality has been widely studied in process calculi [DK04, PU07b, LMS16, CKVI3, MPPY18], event structures [PU15, CKVI5, UY18, GPY18], and lately in Petri nets [BKMP18, PP18, BGM+18]. Despite being a very basic model of concurrency, previous works on Petri nets only provide causally-consistent reversible semantics for subclasses of Petri nets (e.g., acyclic nets or backward-concurrent-free).

A key point when reversing computations in Petri nets is to handle backward conflicts, namely that a token can be generated in a place due to different causes. Consider the net in Figure 1(a) showing the initial state of a system that can either perform \( t_1 \) followed by \( t_3 \), or \( t_2 \) followed by \( t_4 \). The final state of a complete computation is depicted in Figure 1(b). The information in that state is not enough to deduce whether the token in \( d \) has been produced because of \( t_3 \) or \( t_4 \). Even worse, if we “naively” reverse the net by just adding transitions in the reverse direction, as shown in Figure 1(c) (where reverse firings are represented by dashed arrows and the names of their transitions are decorated with `−→` as in, for example, \( t_1 \)), the reverse transitions will do more than just undoing the computation. In fact, the token in \( d \) can be put back either in \( b \) or in \( c \) regardless of which place it came from.

Analogous problems arise when a net is cyclic. Previous approaches [PP18, BGM+18] to reversing Petri nets tackle backward conflicts by relying on a new kind of tokens, called bonds, that keep track of the execution history. Bonds are rich enough for allowing other approaches to reversibility such as out-of-causal order reversibility [PUY13, KU16, KU18], however they
We base our paper on the following simple question:

*Can we define a reversible semantics for Petri nets by relying on standard notions of Petri net theory?*

We provide here an affirmative answer to this question. We propose a reversible model for p/t nets that can handle cyclic nets by relying on standard notions in Petri net theory. We first observe that a Petri net can be mapped via the standard unfolding construction to an occurrence net, i.e., an acyclic net that does not have backward conflicts and makes causal dependencies explicit. Then, an occurrence net can be reversed by adding a reverse transition for each of its transitions. Such construction gives a model where causal-consistent reversibility holds (shown in Section 4). We also prove that each reachable marking in the reversible version of an occurrence net can be reached by just forward computational steps. We observe that the unfolding construction could produce an infinite occurrence net. However, the unfolding can be seen as the definition of a coloured net, where colours account for causal histories. Such interpretation associates a p/t net with an equivalent coloured p/t net, which can be reversed correspondingly as described above. The correctness of the construction is shown by exhibiting a one-to-one correspondence of its executions with the ones of the reversible version of the unfolding. Interestingly, the colours used by the construction resemble the memories common in reversible calculi [DK04, LMSS11, LMS16].

We remark that our proposal deals with reversing (undoing) computations in a Petri net and not with the classical problem of reversibility [CLM76] which requires every computation to be able to reach back the initial state of the system (but not necessary by undoing the previous events). In the latter case, the problem of making a net reversible equates to adding a minimal amount of transitions that make a net reversible [BKMP18, ML19]. Reversibility is a global property while reversing a computation is a local one, as discussed in [BKMP18, ML19].

This paper is an extended version of [MMU19], which includes proofs of our results and many additional examples and explanations.

## 2. Background

### 2.1. Petri Nets

Petri nets are built up from *places* (denoting, e.g., resources and message types), which are repositories for *tokens* (representing instances of resources), and *transitions*, which fetch and produce tokens. We consider the infinite sets \( \mathcal{P} \) of places and \( \mathcal{T} \) of transitions, and assume that they are disjoint, i.e., \( \mathcal{P} \cap \mathcal{T} = \emptyset \). We let \( a, a', \ldots \) range over \( \mathcal{P} \) and \( t, t', \ldots \) over \( \mathcal{T} \). We write \( x, y, \ldots \) for elements in \( \mathcal{P} \cup \mathcal{T} \).

A *multiset* over a set \( S \) is a function \( m : S \to \mathbb{N} \) (where \( \mathbb{N} \) denotes the natural numbers including zero). We write \( \mathbb{N}^S \) for the set of multisets over \( S \). For \( m \in \mathbb{N}^S \), \( \text{supp}(m) = \{ x \in S \mid m(x) > 0 \} \) is the *support* of \( m \), and \( |m| = \sum_{x \in S} m(x) \) stands for its *cardinality*. We write \( \emptyset \) for the empty multiset, i.e., \( \text{supp}(\emptyset) = \emptyset \). The union of \( m_1, m_2 \in \mathbb{N}^S \), written \( (m_1 \oplus m_2) \), is defined such that \( (m_1 \oplus m_2)(x) = m_1(x) + m_2(x) \) for all \( x \in S \). Note that \( \oplus \) is associative and commutative, and has \( \emptyset \) as identity. Hence, \( \mathbb{N}^S \) is the free commutative monoid \( S^\oplus \) over \( S \). We write \( x \) for a singleton multiset, i.e., \( \text{supp}(x) = \{ x \} \) and \( m(x) = 1 \). Moreover, we write \( x_1 \ldots x_n \) for \( x_1 \oplus \ldots \oplus x_n \). Let \( f : S \to S' \), we write \( f \) also for its obvious extension to multisets, i.e., \( f(x_0 \ldots x_n) = f(x_0) \ldots f(x_n) \). We avoid
writing $\text{supp}(\cdot)$ when applying set operators to multisets, e.g., we write $x \in m$ or $m_1 \cap m_2$ instead of $x \in \text{supp}(m)$ or $\text{supp}(m_1) \cap \text{supp}(m_2)$.

**Definition 2.1 (Petri Net).** A net $N$ is a 4-tuple $N = (S_N, T_N, \cdot_N, \cdot_N)$ where $S_N \subseteq \mathcal{P}$ is the (nonempty) set of places, $T_N \subseteq \mathcal{T}$ is the set of transitions and the functions $\cdot_N, \cdot_N : T_N \to 2^{S_N}$ assign source and target to each transition such that $\cdot t \neq \emptyset$ and $\cdot t \neq \emptyset$ for all $t \in T_N$. A marking of a net $N$ is a multiset over $S_N$, i.e., $m \in \mathbb{N}^{S_N}$. A Petri net is a pair $(N, m)$ where $N$ is a net and $m$ is a marking of $N$.

We denote $S_N \cup T_N$ by $N$, and omit the subscript $N$ if no confusion arises. We abbreviate a transition $t \in T$ with preset $\cdot t = s_1$ and postset $\cdot t = s_2$ as $s_1 \parallel s_2$. The preset and postset of a place $a \in S$ are defined respectively as $\cdot a = \{ t | a \in t \}$ and $a^* = \{ t | a \in t \}$. We let $\circ N = \{ x \in N | \cdot x = \emptyset \}$ and $\circ^o N = \{ x \in N | a^* = \emptyset \}$ denote the sets of initial and final elements of $N$ respectively. Note that we only consider nets whose initial and final elements are places since transitions have non-empty presets and postsets, namely $\cdot t \neq \emptyset$ and $\cdot t \neq \emptyset$ hold for all $t$.

**Definition 2.2 (Net morphisms).** Let $N, N'$ be nets. A pair $f = (f_S : S_N \to S_{N'}, f_T : T_N \to T_{N'})$ is a net morphism from $N$ to $N'$ (written $f : N \to N'$) if $f_S(\cdot_N t) = \cdot_{N'} (f_T t)$ and $f_S(t_N) = (f_T(t))_{N'}$ for any $t$. Moreover, we say $N$ and $N'$ are isomorphic if $f$ is bijective.

The operational (interleaving) semantics of a Petri net is given by the least relation on Petri nets satisfying the following inference rule:

\[
\frac{t = m \parallel m' \in T_N}{(N, m \oplus m') \overset{t}{\to} (N, m' \oplus m'')}\]

which describes the evolution of the state of a net (represented by the marking $m \oplus m''$) by the firing of a transition $m \parallel m'$ that consumes the tokens $m$ in its preset and produces the tokens $m'$ in its postset. We shall call expressions of the form $(N, m \oplus m'') \overset{t}{\to} (N, m' \oplus m'')$ as firings. We sometimes omit $t$ in $\overset{t}{\to}$ when the fired transition is uninteresting.

According to Definition 2.1, each transition consumes and produces at most one token in each place. On the other hand, $p/t$ nets below fetch and consume multiple tokens by defining the pre- and postsets of transitions as multisets.

**Definition 2.3 (p/t net).** A Place/Transition Petri net $(p/t)$ net is a 4-tuple $N = (S_N, T_N, \cdot_N, \cdot_N)$ where $S_N \subseteq \mathcal{P}$ is the (nonempty) set of places, $T_N \subseteq \mathcal{T}$ is the set of transitions and the functions $\cdot_N, \cdot_N : T_N \to \mathbb{N}^{S_N}$ assign source and target to each transition such that $\cdot t \neq \emptyset$ and $\cdot t \neq \emptyset$ for all $t \in T_N$. A marking of a net $N$ is multiset over $S_N$, i.e., $m \in \mathbb{N}^{S_N}$. A marked $p/t$ net is a pair $(N, m)$ where $N$ is a $p/t$ net and $m$ is a marking of $N$.

The notions of pre- and postset, initial and final elements, morphisms and operational semantics are straightforwardly extended to $p/t$ nets. For technical reasons, we only consider $p/t$ nets whose transitions have non-empty pre- and post-sets. Note that Petri nets can be regarded as a $p/t$ net whose arcs have unary weights.

Next, we introduce notation for sequences of firings (or transitions). Let $\cdot$ denote concatenation of such sequences. For the sequence $s = t_1; t_2; \ldots; t_n$, we write $(N, m_0) \overset{t_1}{\to} (N, m_1) \overset{t_2}{\to} \ldots \overset{t_n}{\to} (N, m_n)$; we call $s$ a firing sequence. We write
\((N, m_0) \rightarrow^* (N, m_n)\) if there exists \(s\) such that \((N, m_0) \xrightarrow{s} (N, m_n)\), and \(\epsilon_m\) for the empty sequence with the marking \(m\).

**Definition 2.4.** Let \((N, m)\) be a P/T net. The set of reachable markings \(\text{reach}(N, m)\) is defined as \(\{m' \mid (N, m) \rightarrow^* (N, m')\}\).

We say a marked P/T net \((N, m)\) is (1-)safe if every reachable marking is a set, i.e., \(m' \in \text{reach}(N, m)\) implies \(m' \in 2^{SN}\).

**Example 2.5.** Figure 2 shows different P/T nets, which will be used throughout the paper. As usual, places and transitions are represented by circles and boxes, respectively. The nets \(O_1\) and \(N_4\) are Petri nets, and \(N_1, N_2, N_3\) and \(N_5\) are P/T nets which, when executing, may produce multiple tokens in some places.

### 2.2. Unfolding of P/T nets

Our approach to reversing Petri nets relies on their occurrence net semantics, which explicitly exhibits the causal ordering, concurrency, and conflict among places and transitions. We start by introducing several useful notions and notations. First, we shall describe a flow of causal dependencies in a net with the relation \(\prec\):

**Definition 2.6.** Let \(\prec\) be \(\{(a, t) \mid a \in SN \land t \in a^*\} \cup \{(t, a) \mid a \in SN \land t \in a^*\}\). We write \(\preceq\) for the reflexive and transitive closure of \(\prec\).

Consider Figure 2. We have \(a \prec t_1\) and \(t_1 \prec c\) in \(O_1\) as well as \(t_1 \preceq t_2\) in \(N_1\).
Two transitions \( t_1 \) and \( t_2 \) are in an immediate conflict, written \( t_1 \#_0 t_2 \), when \( t_1 \neq t_2 \) and \( \bullet t_1 \cap \bullet t_2 \neq \emptyset \). For example, \( t_2 \) and \( t_3 \) in \( N_1 \) in Figure 2 are in an immediate conflict since they share a token in the place \( c \). Correspondingly, for \( t_1 \) and \( t_2 \) in \( N_4 \). The conflict relation \# is defined by letting \( x \# y \) if there are \( t_1, t_2 \in T \) such that \( t_1 \preceq x \), and \( t_2 \succeq y \), and \( t_1 \#_0 t_2 \).

We are now ready to define occurrence nets following [NPW81, HW08].

**Definition 2.7** (Occurrence net). A net \((N, m)\) is an occurrence net if

1. \( N \) is acyclic;
2. \( N \) is a \((1-)\)-safe net, i.e., any reachable marking is a set;
3. \( m = \circ N \), i.e., the initial marking is identified with the set of initial places;
4. there are no backward conflicts, i.e., \( |\bullet a| \leq 1 \) for all \( a \) in \( S_N \);
5. there are no self-conflicts, i.e, \( \neg t \# t \) for all \( t \) in \( T_N \).

We use \( O, O', \ldots \) to range over occurrence nets.

**Example 2.8.** The net \( O_1 \) in Figure 2 is an occurrence net, while the remaining nets are not. \( N_1 \) is not an occurrence net since there is a token in place \( c \) and \( c \) is not an initial place of the net. \( N_2 \) has a backward conflict since two transitions produce tokens on the place \( d \). \( N_3 \) is cyclic, and \( N_4 \) is cyclic and has a backward conflict on \( c \). \( N_5 \) is not an occurrence net since it is not \( 1\)-safe.

The absence of backward conflicts in occurrence nets ensures that each place appears in the postset of at most one transition. Hence, pre- and postset relations can be interpreted as a causal dependency. So, \( \preceq \) represents causality.

We say \( x, y \in S_N \cup T_N \) are concurrent, written \( x \text{ co } y \), if \( x \neq y \) and \( x \not\preceq y \), \( y \not\preceq x \), and \( \neg x \# y \). A set \( X \subseteq S_N \cup T_N \) is concurrent, written \( CO(X) \), if \( \forall x, y \in X : x \neq y \Rightarrow x \text{ co } y \), and \( |\{ t \in T_N \mid \exists x \in X, t \preceq x \}| \) is finite. For example, the set \( \{ t_1, t_2 \} \) of firings in \( O_1 \) of Figure 2 is concurrent, so we can write \( CO(\{ t_1, t_2 \}) \).

The notions of causality, conflict and concurrence for firings are defined in terms of the corresponding notions for their transitions. For example, the firing \( (N, m) \xrightarrow{\bullet} (N, m') \) causes the firing \( (N, m_1) \xrightarrow{t'} (N, m_1') \), written as \( (N, m) \xrightarrow{t} (N, m_1) \preceq (N, m_1') \), if \( t \preceq t' \). Correspondingly for the conflict and concurrency relations.

Two firings (or transitions) are coinital if they start with the same marking, and cofinal if they end up in the same marking. We now have a simple version of Square Lemma [DK04] for forward concurrent firings. It will be helpful in proving our Lemma 4.3 in Section 4.

**Lemma 2.9.** Let \( O \) be an occurrence net and \( t \) and \( t' \) be enabled cofinal transitions of \( O \). Then \( t \text{ co } t' \) if and only if \( \bullet t \cap \bullet t' = \emptyset \).

**Proof.** \( \Rightarrow \) part: \( t \text{ co } t' \) implies \( \neg t \#_0 t' \). Hence, \( \bullet t \cap \bullet t' = \emptyset \).

\( \Leftarrow \) part: We proceed by contradiction. Assume that \( t \) and \( t' \) are enabled at some marking \( m \) of \( O \) and \( \neg (t \text{ co } t') \). Since \( \bullet t \cap \bullet t' = \emptyset \), either \( t \preceq t' \) or \( t' \preceq t \) holds (note that \( t \preceq t' \) and \( t' \preceq t \) do not hold simultaneously because occurrence nets are acyclic). We consider the case \( t \preceq t' \) and show that \( t \) is enabled implies that \( t' \) is not enabled. Since, \( t \preceq t' \), there exists \( a \in \bullet t' \) such that \( t \prec a \). Therefore, \( a \not\in \circ O \). Finally, \( |\bullet a| \leq 1 \) and \( t \prec a \) imply \( a \not\in m \), which is in contradiction with the assumption that \( t' \) is enabled at \( m \). The case \( t' \preceq t \) follows analogously. \( \square \)
Lemma 2.10. Let \( O \) be an occurrence net and \( (O, n) \xrightarrow{s} (O, n') \) and \( (O, n) \xrightarrow{t'} (O, n'') \) be coinital concurrent firings. Then, there exist firings \( (O, n') \xrightarrow{t'} (O, n_1') \) and \( (O, n'') \xrightarrow{t'} (O, n_1'') \), for some \( n_1' \) and \( n_1'' \), such that the firings are cofinal.

Proof. Assume that \( (O, n) \xrightarrow{s} (O, n') \) and \( (O, n) \xrightarrow{t'} (O, n'') \) are coinital concurrent firings. That means \( t \) and \( t' \) are coinital, concurrent and are enabled. Hence they do not cause each other, and they are also not in an immediate conflict. So, we can write the firings as \( (O, m \oplus m' \oplus m'') \xrightarrow{s'} (O, m_1 \oplus m' \oplus m'') \) and \( (O, m \oplus m' \oplus m'') \xrightarrow{t'} (O, m \oplus m_1' \oplus m'') \), where \( t \leq m \) and \( t' \leq m' \) with markings \( m \) and \( m' \) not overlapping. Clearly then these are valid firings: \( (O, m_1 \oplus m' \oplus m'') \xrightarrow{t'} (O, m_1 \oplus m_1' \oplus m'') \) and \( (O, m \oplus m_1' \oplus m'') \xrightarrow{t'} (O, m_1 \oplus m_1' \oplus m'') \). They are cofinal since they end up in \( (O, m_1 \oplus m_1' \oplus m'') \).

This lemma can be equivalently expressed as follows: if \( t \) and \( t' \) are coinital and concurrent, then \( t; t' \) and \( t'; t \) are cofinal. Informally, if firings with transitions \( t \) and \( t' \) originate from one corner of a square, and if they represent independent (concurrent) events, then the square completes with two firings \( t' \) and \( t \), which meet at the opposite corner of the square. Hence, the order in which concurrent transitions are executed in a firing sequence does not matter. We then consider sequences equivalent up to the swapping of concurrent transitions. This corresponds to considering the set of Mazurkiewicz traces induced by \( \text{co} \) as the independence relation.

Formally, trace equivalence \( \equiv \) is the least congruence over firing sequences \( s \) such that \( \forall t_1, t_2 : t_1 \text{ co } t_2 \implies t_1 ; t_2 \equiv t_2 ; t_1 \). The equivalence classes of \( \equiv \) are the (Mazurkiewicz) traces. We use \( \omega \) to range over such traces. We also will use \( \epsilon \) for the empty trace, and ; for the concatenation operator.

For occurrence nets we have this standard property:

\[
s_1 \equiv s_2 \iff (O, m_0) \xrightarrow{s_1} (O, m_n) \iff (O, m_0) \xrightarrow{s_2} (O, m_n)
\]

Two traces are coinital if they start with the same marking, and cofinal if they end up in the same marking. Hence, the above property tells us that two traces that are coinital and cofinal are precisely trace equivalent.

The unfolding of a net \( N \) is the least occurrence net that can account for all the possible computations of \( N \) and makes explicit causal dependencies, conflicts and concurrency between firings [NPW81].
Example 2.12. The unfolding of the net \((N, m)\) in Figure 2 is given in Figure 4.

Henceforth, we adopt the usual convention of omitting causal histories in the name of places and transitions when depicting unfoldings, and can write instead the image of the unfolding morphism, as illustrated by the following example.

**Definition 2.11** (Unfolding). Let \((N, m)\) be a P/T net. The unfolding of \(N\) is the occurrence net \(\mathcal{U}[N, m] = (S, T, \bullet, \cdot)\) generated inductively by the inference rules in Figure 3, and the folding morphism \((f_S, f_T) : \mathcal{U}[N, m] \to N\) is defined such that \(f_S(a(\cdot, \cdot)) = a\) and \(f_T(t(\cdot)) = t\).

Places in the unfolding of a net represent tokens and are named by triples \(a(H, i)\) where: \(a\) is the place of the original net \(N\) in which the token resides; \(H\) is the set of its immediate causes (i.e., the causal history of the token); and \(i\) is a positive integer used to disambiguate tokens with the same history, i.e., when the initial marking assigns several tokens to a place or a transition produces multiple tokens in a place. Analogously, transitions in the unfolding represent firings or events and are named by pairs \(t(H)\), where \(H\) encodes the causal history as above and \(t\) is the fired transition.

**Example 2.13.** The unfoldings of the nets \((N_1, a \oplus b \oplus c \oplus d)\), \((N_2, a \oplus b \oplus c)\), \((N_3, a)\), \((N_4, a \oplus b \oplus c)\) and \((N_4, a \oplus a \oplus b)\) in Figure 2 are shown in Figure 5. Since \(O_1\) in Figure 2 is an occurrence net its unfolding is isomorphic to \(O_1\), thus it is omitted. Consider the occurrence net \(\mathcal{U}[N_1, a \oplus b \oplus c \oplus d]\). The leftmost transition \(t_2\) is different from the other transition \(t_2\) since they have different histories: the leftmost \(t_2\) is caused by the tokens in \(b\) and \(c\) (which are available in the initial marking), whereas the other \(t_2\) is caused only by the token in \(b\) and the token that is produced by the firing of \(t_1\). Correspondingly, for the two transitions labelled \(t_3\). Consider \(\mathcal{U}[N_2, a \oplus b \oplus c]\). After the transitions \(t_1\) and \(t_2\) have fired, there is a token in each of the places labelled \(d\). The token in the leftmost \(d\) has the history corresponding to the firing of \(t_1\) and the token in the other \(d\) has the history corresponding...
to \( t_2 \). Once \( t_3 \) has fired, we can tell the copies of \( t_3 \) apart by inspecting their histories: the leftmost \( t_3 \) is caused by a token in \( d \) with the history \( t_1 \) (as well as the token in \( c \)), whereas the other \( t_3 \) is caused by \( d \) with the history \( t_2 \) and by \( c \).

### 3. Reversing Occurrence Nets

**Definition 3.1.** Let \( O \) be an occurrence net. The reversible version of \( O \) is \( \overrightarrow{O} = (S_{\overrightarrow{O}}, T_{\overrightarrow{O}}, \cdot_{\overrightarrow{O}}, \overleftarrow{O}) \) defined as follows:

\[
\overrightarrow{O} = (S_{\overrightarrow{O}}, T_{\overrightarrow{O}}, \cdot_{\overrightarrow{O}}, \overleftarrow{O})
\]

- \( S_{\overrightarrow{O}} = S_O \)
- \( T_{\overrightarrow{O}} = T_O \cup \{ \overleftarrow{t} | t \in T_O \} \)
- \( \cdot_{\overrightarrow{O}} = \begin{cases} \cdot_O & \text{if } t \in T_O \\ \cdot_O & \text{otherwise} \end{cases} \)
- \( \overleftarrow{O} = \begin{cases} \overleftarrow{t} & \text{if } t \in T_O \\ \overleftarrow{t} & \text{otherwise} \end{cases} \)

Given a transition \( t \) we write \( \overleftarrow{t} \) for a transition that reverses \( t \). We shall call transitions like \( \overleftarrow{t_1} \) and \( \overleftarrow{t_2} \) in Figure 6 reverse (or backwards) transitions and transitions like \( t \) forward transitions. We will use \( t, t_1, t_2, \ldots \) to denote forward or reverse transitions. If \( t \) is a reverse transition, say \( \overleftarrow{t} \), then \( \overleftarrow{t} \) is the forward transition \( t \).
Example 3.2. The reversible version of the nets in Figure 2 are shown in Figure 6. We remark that they are the reversible versions of the nets in Figure 5, which are the unfoldings of the original nets.

Figure 6: Reversible P/T and Petri nets
Given \( \vec{O} \), we write \((\vec{O}, m) \xrightarrow{t} (\vec{O}, m')\) for a forward firing when \( t \in T_O \), and \((\vec{O}, m) \xrightarrow{t}^\leftarrow (\vec{O}, m')\) for the reverse (or backward) firing when \( t \notin T_O \). We let \( \xrightarrow{t} \) be \( \xrightarrow{t} \cup \xrightarrow{t}^\leftarrow \). Henceforth we often refer to firings \((\vec{O}, m) \xrightarrow{t} (\vec{O}, m')\), \((\vec{O}, m) \xrightarrow{t}^\leftarrow (\vec{O}, m')\) and \((\vec{O}, m) \xrightarrow{T} (\vec{O}, m')\) simply as \( t \), \( t \) and \( \vec{T} \) respectively, especially when discussing properties of \( \xrightarrow{t} \) in this section and in Section 4. We shall work with sequences of firings, ranged over by \( s, s_1 \) and \( s_2 \). We say that a sequence is a forward (respectively backward) sequence when all its firings are forward (respectively backward).

Next, we extend the notions of causality, conflict and concurrency to forward transitions and reverse transitions in reverse versions of occurrence nets. We extend \( \prec \) in Definition 2.6 to cover reverse transitions in an obvious way using Definition 3.1. As a result, we obtain \( t \leq \vec{t} \) and \( \vec{T} \leq t \). We note that although \( \leq \) is now circular, there are no loops of forward transitions only, and all loops involve forward transitions followed by their reverse versions (in the inverse order).

The conflict relation for forward transitions or reverse transitions is defined correspondingly as in Section 2. There is an immediate conflict between \( t \) and \( t' \), written as \( t \nmid t' \), if \( \bullet t \cap \bullet t' \neq \emptyset \). When \( t \) and \( t' \) are forward transitions, then \( t \nmid t' \) is as in Section 2.2. If \( t \) is \( t \) and \( t' \) is \( \vec{t} \), then \( t \nmid t' \) is \( \bullet t \cap \bullet \vec{t} \neq \emptyset \), which is equivalent to \( \bullet t \cap t' \neq \emptyset \), that trivially holds in occurrence nets. Hence, the immediate conflict relation is empty between reverse transitions. The conflict relation \( \# \) between \( t \) and \( t' \) is as in in Section 2.2: \( t \nmid t' \) if there are forward transitions \( t_1, t_1' \in T_O \) such that \( t_1 \preceq t \), \( t_1' \preceq t' \), and \( t_1 \nmid t_1' \).

Having introduced the notions of causality and conflict, we can now define concurrent transitions in \( \vec{O} \). We follow the definition in Section 2.2: \( t, t' \) are concurrent, written \( t \co t' \), if \( t \neq t' \) and \( t \npreceq t' \), \( t' \npreceq t \), and \( \neg t \nmid t' \).

**Lemma 3.3.** Let \( t \) and \( t' \) be enabled coinitial forward or reverse transitions of a reversible occurrence net. Then \( t \co t' \) if and only if (a) \( \neg(t \nmid t') \) if \( t, t' \) are forward transitions, (b) \( t \npreceq t' \) and \( t' \npreceq t \) if \( t, t' \) are reverse transitions, and (c) \( t' \npreceq t \) and \( \vec{t} \npreceq t \) if \( t \) is a forward transition and \( t' \) is a reverse transition.

**Proof.** \( \Rightarrow \) part: This is obvious by the definition of \( \co \) and Lemma 2.9.

\( \Leftarrow \) part: We have three cases. If \( t \) and \( t' \) are forward transitions, then we are done by Lemma 2.9. If \( t, t' \) are reverse transitions, then since they are initial and enabled, they cannot be in conflict unless they are in an immediate conflict. However, an immediate conflict of two reverse transitions is equivalent to a backward conflict of their forward versions, which is not allowed in occurrence nets. Hence \( t \co t' \) as long as they do not cause each other. Finally, consider a forward transition \( t \) and a reverse transition \( t' \). Then \( t \npreceq t' \), one of the conditions for \( t \co t' \), is equivalent to no backward conflict between \( t \) and \( \vec{t} \), which is guaranteed in occurrence nets. No immediate conflict between \( t \) and \( t' \) is equivalent to \( \vec{t} \npreceq t \) and, with \( t' \npreceq t \) given, we obtain \( t \co t' \).

The next result, which follows by the definition of the concurrency relation on forward and reverse transitions, will be helpful in the next section when we consider consecutive concurrent transitions.

**Lemma 3.4.** Let \( t, t' \) be a trace of a reversible occurrence net. Then, \( t \co t' \) if and only if \( \vec{t} \co t' \).
Definition 3.1 and the discussion above allow us to show that $\tilde{O}$ is a conservative extension of $O$.

**Lemma 3.5.** $(O, m) \xrightarrow{t} (O, m')$ if and only if $(\tilde{O}, m) \xrightarrow{t} (\tilde{O}, m')$.

In general, a reversible occurrence net is not an occurrence net. This is because adding reverse transitions introduces cycles and backward conflict. Consider $N_1$ in Figure 2. We notice that initially $t_1$ and $t_2$ are in conflict. Then, in $\tilde{N}_1$ in Figure 6, the place $c$ that contains a token has two reverse transitions in its preset, namely $\tilde{t}_2$ and $\tilde{t}_3$, hence there is a backward conflict.

4. Properties

We now study the properties of the reversible versions of occurrence nets. We follow here the approach of Danos and Krivine [DK04] and work with a transition system defined in Section 3. The aim is to show that reversibility in reversible occurrence nets is causal-consistent (Theorem 4.6). Informally, this means that we can reverse a firing (or a transition) in a computation of a concurrent system as long as all firings (transitions) caused by the firing (transition) have been undone first. We shall need several useful properties over firings, transitions or their sequences before we can prove Theorem 4.6.

An important property of a fully reversible system is the lemma below stating that any forward transition can be undone. In the setting of reversible occurrence nets this is stated as follows:

**Lemma 4.1** (Loop Lemma). Let $O$ be an occurrence net. Then, $(\tilde{O}, m) \xrightarrow{t} (\tilde{O}, m')$ if and only if $(\tilde{O}, m') \xleftarrow{t} (\tilde{O}, m)$.

We can generalise this property to sequences of transitions and reverse transitions as follows:

**Corollary 4.2.** Let $O$ be an occurrence net. Then, $(\tilde{O}, m) \rightarrow^\ast (\tilde{O}, m')$ if and only if $(\tilde{O}, m') \rightarrow^\ast (\tilde{O}, m)$.

Next, we have a lemma which is instrumental in the proof of causal-consistent reversibility [DK04, LMS16]. Note that $t$ and $t'$ can be either forward or reverse transitions.

**Lemma 4.3** (Square Lemma). Let $t$ and $t'$ be enabled coinitial concurrent transitions of a reversible occurrence net. Then, there exist transitions $t_1$ and $t_1'$ such that $t; t_1'$ and $t'; t_1$ are cofinal.

**Proof.** We consider three cases when $t$ and $t'$ are enabled, coinitial and concurrent transitions. If $t$ and $t'$ are both forward then we are done by Lemma 2.10. If $t$ and $t'$ are both reverse, then they cannot cause one another and they cannot be in an immediate conflict (because this would imply a backward conflict on their forward versions which we do not have in occurrence nets: see Lemma 3.3). So, $t, t'$ do not share tokens in their presets. Hence, we can write $t, t'$ correspondingly as we wrote $t, t'$ in the proof of Lemma 2.10. The rest of the case follows correspondingly as in the proof of Lemma 2.10. If $t$ is a transition and $t'$ is a reverse transition, then since they are coinitial, concurrent and enabled they cannot cause one another and they are not in an immediate conflict, which is equivalent to $\tilde{t}' \not\leq t$. Hence the preset of $t$ does not overlap with the preset of $t'$. We then finish as in the proof of Lemma 2.10. \qed
In order to prove causal consistency we first define a notion of equivalence on sequences of forward and reverse transitions in reversible occurrence nets. By following the approach in [Lév76, DK04], we define the notion of reverse equivalence on such sequences as the least equivalence relation $\simeq$ which is closed under composition with ‘;’ such that the following rules hold:

$$t; t' \simeq t; t' \text{ if } t \text{ co } t'$$

$$t; t' \simeq t; t' \text{ if } t \text{ co } t'$$

The reverse equivalence $\simeq$ allows us to swap the order of $t$ and $t'$ in an execution sequence as long as $t$ and $t'$ are concurrent. Moreover, it allows cancellation of a transition and its inverse. We have that $\equiv \subset \simeq$. The equivalence classes of $\simeq$ are called traces; it is clear that they contain the Mazurkiewicz traces. Hence, we shall use $\omega, \omega_1, \omega_2$ to range over such traces.

The following lemma says that, up to reverse equivalence, one can always reach for the maximum freedom of choice, going backwards first and only then going forwards.

**Lemma 4.4 (Parabolic Lemma).** Let $\omega$ be a trace of a reversible occurrence net. There exist two forward traces $\omega_1$ and $\omega_2$ such that $\omega \simeq \omega_1; \omega_2$.

**Proof.** The proof is by lexicographic induction on the length of $\omega$ and on the distance between the beginning of $\omega$ and the earliest pair of transitions in $\omega$ of the form $t_1; t_2$, where $t_1$ and $t_2$ are forward. If such pair does not exist, then the result follows immediately (i.e., either $\omega_1 = \epsilon$ or $\omega_2 = \epsilon$). If there exists one such pair we have two possibilities: either (i) $t_1$ and $t_2$ are concurrent or (ii) they are not. Case (i) also implies that $\overleftarrow{t_1}$ and $\overleftarrow{t_2}$ are coinitial, enabled and also concurrent by Lemma 3.4. Lemma 4.3 implies that there are cofinal firings with labels $\overleftarrow{t_2}$ and $\overleftarrow{t_1}$ that making up a square. In addition to the subtrace $t_1; t_2$, there is a coinitial and cofinal subtrace $t_2; t_1$ on the other side of the square. So $t_1$ and $t_2$ can be swapped, obtaining a later earliest pair of the form $t; t'$. The result then follows by induction since swapping keeps the total length of the trace unchanged.

In case (ii), $t_1$ and $t_2$ are not concurrent. Then, we have the following four cases:

1. $t_1 \succeq \overleftarrow{t_2}$. This implies that $t_1 \cap \overleftarrow{t_2} \neq \emptyset$ which is equivalent to $t_1 \cap t_2 \neq \emptyset$. This means that there is a backward conflict between $t_1$ and $t_2$. Since occurrence nets are free of backward conflict we get a contradiction.

2. $\overleftarrow{t_2} \succeq t_1$. This implies that $t_2 \cap t_1 \neq \emptyset$, and that $t_2 \cap t_1 \neq \emptyset$. The last says that $t_1$ and $t_2$ are in an immediate conflict. Since we have $t_1; t_2$ in $\omega$, and all transitions before $t_1$ in $\omega$ are forward transitions, it means that to undo $t_2$ the transition $t_2$ must have occurred before $t_1$. However, this contradicts $t_1$ and $t_2$ being in an immediate conflict, which requires that if one transition takes place then the other cannot happen as they share tokens in their preset and there are no cycles among the forward transitions.

3. $t_1 \#_0 t_2$. By definition of $\#_0$ we have $t_1 \cap t_2 \neq \emptyset$. Since $t_1; t_2$ appears in $\omega$, $t_2$ should remain enabled after the firing of $t_1$. Since occurrence nets are 1-safe and acyclic, this contradict the assumption $t_1 \#_0 t_2$.

4. $t_1 \#_0 t_2$. This implies that there exist earlier $t$ and $t'$ in $\omega$ such that $t \succeq t_1$ and $t' \succeq t_2$ with $t \#_0 t'$. So we have $t$ and $t'$ in an immediate conflict appearing in $\omega$ prior to sub-trace $t_1; t_2$. We then show a contradiction as in case (3) above.

\footnote{Note that it is not possible that some transition $t_3$ has produced a token to a place $a$ that was used by $t_2$: this would imply $\{t_2, t_3\} = \{a\}$, which is a backward conflict: contradiction.}
The following lemma says that, if two traces $\omega_1$ and $\omega_2$ are coinital and cofinal (namely they start from the same marking and end in the same marking) and if $\omega_2$ has only forward transitions, then $\omega_1$ has some forward transitions and their reverse versions that can cancel each other out. A consequence of this is that overall $\omega_1$ is causally equivalent to a forward trace $\omega'_1$ in which all pairs of inverse transitions are cancelled out.

**Lemma 4.5** (Shortening Lemma). Assume $\omega_1$ and $\omega_2$ are coinital and cofinal traces, and $\omega_2$ is forward. Then, there exists a forward trace $\omega'_1$ with $|\omega'_1| \leq |\omega_1|$ such that $\omega'_1 \equiv \omega_1$.

**Proof.** By induction on the length of $\omega_1$. If $\omega_1$ is a forward trace, then we are already done. Otherwise, by applying Lemma 4.4, we have that $\omega_1 \equiv \omega''$ where both $\omega$ and $\omega'$ are forward traces. Consider the sub-trace $t_1 ; t_2$ in $\omega''$. Clearly $t_1 ; t_2$ is the only sub-trace of this form in $\omega''$. Suppose $t_1 = m'$ and $t_2 = m''$. Since $\omega_2$ is a forward only trace and since $\omega_1(\equiv \omega'' \equiv \omega')$ and $\omega_2$ coinital and cofinal, the marking $m''$ removed by $t_1$ has to be put back in $\omega_1$, otherwise this change would be visible in $\omega_2$ (since it is a forward only trace). Let $t_3$ the earliest transition in $\omega'$ able to consume $m''$ and put back $m'$. This implies that $t_3 = t_1$ since there are no loops of forward transitions in occurrence nets. If there are no forward transitions between $t_1$ and $t_3$, then we can cancel them out by applying the rules for $\equiv$. Then, we are done by induction on a shorter trace.

Otherwise, assume there are forward transitions between $t_1$ and $t_4$, and $t_4$ is the last such transition. Next, we show that $t_4$ and $t_1$ are concurrent so that they can be swapped thus moving $t_1$ closer to $t_1$. It is sufficient to show $t_4$ and $t_1$ are concurrent by Lemma 3.4. So we require $t_4 \not\equiv t_1$ and $t_4 \not\equiv t_1$ by Lemma 3.3. Since $t_4 ; t_1$ is a trace, the transitions cannot be in an immediate conflict. This is equivalent to $t_4 \cap t_1 = \emptyset$, which is $t_4 \not\equiv t_1$. So, we are only left to prove $t_4 \not\equiv t_1$. We proceed by contradiction. Assume that $t_4 \equiv t_1$. Hence, $t_4 \cap t_1 \not= \emptyset$, so $a \in t_4 \cap t_1$ for some place $a$. After $t_1$ takes place in $\omega_1$ there will be a token in $a$. If that token stays there while computation progresses towards $t_4$, then $t_4$ will place another token in $a$: contradiction with the 1-safe property. So, there is a transition $t_5$ between $t_1$ and $t_4$ that consumes the token from $a$. Hence, $t_5$ and $t_1$ are in an immediate conflict and form a forward trace $t_5 ; \ldots ; t_4 ; t_1$: contradiction.

Since $t_4$ and $t_3$ are concurrent, so are $t_4$ and $t_1$. Hence, $t_4 ; t_1 \equiv t_1 ; t_4$, which moves $t_1$ closer to $t_1$. We then continue this way until transitions $t_1$ and $t_1$ are adjacent. Finally, we cancel them out by applying the rules for $\equiv$ and then we conclude by induction on a shorter trace.

Next, we give our causal consistency result. If two traces composed of transitions or reverse transitions are coinital and cofinal, then they are equivalent with respect to $\equiv$, and vice versa. Given a computation represented by trace $\omega$ we can reverse it by simply doing $\omega$, which would be backtracking, or by doing $\omega^r$ if $\omega \equiv \omega^r$. The latter option allows us to undo transitions in any order as long as all the consequences of these transitions have been undone first.

**Theorem 4.6** (Causal Consistency). Let $\omega_1$ and $\omega_2$ be two coinital traces. Then, $\omega_1 \equiv \omega_2$ if and only if $\omega_1$ and $\omega_2$ are cofinal.

**Proof.** $\Rightarrow$ part: If $\omega_1$ and $\omega_2$ are coinital and $\omega_1 \equiv \omega_2$, then $\omega_1$ and $\omega_2$ are cofinal. We notice that if $\omega_1 \equiv \omega_2$, then $\omega_1$ can be transformed into $\omega_2$ (and vice versa) via $n \geq 0$ applications of the rules of $\equiv$. So we proceed by induction on $n$. For $n = 0$ we have that $\omega_1 \equiv \omega_2$ by
applying 0 times the rules of ∞. Since ∞ is an equivalence, this means that ω1 = ω2 which in turn implies that the traces are coinitial and cofinal. In the inductive case there exist n traces ωk (with 0 ≤ k ≤ n) obtained as a result of applying the rules of ∞ to ω1 exactly k times; hence, ω0 = ω1 and ωn = ω2. We then have ω1 ∼ ωn−1, and ωn−1 ∼ ω2, i.e., traces ωn−1 and ω2 differ in one rule application. This means that we can decompose both traces as ωn−1 = ω0; ω′ and ω2 = ω0; ω′′, where ω′ and ω′′ are shortest traces that differ by just one application of the rules for ∞. Then, there are three cases:

1. ω′ = t1; t2 and ω′′ = t2; t1 with t1 co t2;
2. ω′ = t; t and ω′′ = ϵm with m the marking before firing t;
3. ω′ = t; t and ω′′ = ϵm with m the marking after firing t.

We need to show that ω1 and ω2 are cofinal. In all the cases above it is easy to see that ωn−1 and ω2 are both coinitial and cofinal. By the inductive hypothesis, ω1 ∼ ωn−1 implies that ω1 and ωn−1 are coinitial and cofinal. Hence, we can conclude that also ω1 and ω2 are coinitial and cofinal.

⇐ part: If ω1 and ω2 are coinitial and cofinal, then ω1 ∼ ω2. We can assume by Lemma 4.4 that ω1 and ω2 are compositions of a backward trace and a forward trace. The proof is by lexicographic induction on the sum of the lengths of ω1 and ω2, and on the distance between the end of ω1 and t1 of the earliest pair of transitions that do not agree is closer to the end of the trace, ω1 and ω2 are equal then we are done. Otherwise, we have to consider three cases depending on the direction of the two transitions.

1. t1 = t1 and t2 = t2. We have ω1 = t1; t2; ω′ and ω2 = t1; t2; ω′′, where ω is the common backward sub-trace and t1; ω′ is a forward trace. Since ω1 and ω2 are coinitial and cofinal, this implies that also t1; ω′ and t2; ω′′ are coinitial and cofinal. By applying Lemma 4.5 to the trace t2; ω′′ we obtain a shorter equivalent forward trace, and therefore ω2 has a shorter causally equivalent trace. We then conclude by induction.

2. t1 = t1 and t2 = t2. By assumption we have t1 ̸= t2. We can decompose the two traces as follows: ω1 = ω; t1; ω1 and ω2 = ω; t2; ω2, for some ω, ω1 and ω2, where all the transitions in ω1 and ω2 are forward. Next, we show that t1 and t2 are concurrent. Assume for contradiction that t1 and t2 are in an immediate conflict. This means *t1 ∩ *t2 = ∅.

Since the two traces are coinitial we can write

\[
(\hat{D}, m_0) \xrightarrow{\omega_1} (\hat{D}, m_1) \xrightarrow{t_1} (\hat{D}, m_f) \]

\[
(\hat{D}, m_0) \xrightarrow{\omega_2} (\hat{D}, m_2) \xrightarrow{t_2} (\hat{D}, m_f)
\]

where m1 = m2 and m_f is the final marking. The effect of t1 remains visible at the end of ω1, namely in m_f, and hence at the end of ω2. This implies that there is a transition in ω2 that produces the tokens necessary to enable t1. Hence, there is a loop. This is a contradiction since here we consider forward transitions only of occurrence nets, which by definition are acyclic with regards to forward transitions. Hence, t1 co t2 by Lemma 3.3, and t1 ∈ ω2. If there are no transitions between t2 and t1 in ω2, then we swap them by applying the rules for ∞ thus obtaining ω2 ∼ ω; t1; t2; ω3, where ω3 is ω2 without the leading t1. The traces ω; t1; ω1 and ω; t1; t2; ω3 have the same length as before, but the first pair of transitions that do not agree is closer to the end of the trace, and the result follows by induction. Otherwise, assume there are forward transitions between t2 and t1, and let t3 be the last such transition. We now show that t1 and
t₃ are concurrent, which requires t₃ ⪯ t₁ and t₃ ⩾ t₁ by Lemma 3.4 and Lemma 3.3. Since t₃; t₁ is a trace, the transitions cannot be in an immediate conflict: \( t₃ \cap t₁ = \emptyset \). This is equivalent to \( t₃ \cap t₁ = \emptyset \), which is t₃ ⩾ t₁. So we only need to prove t₃ ⪯ t₁.

Assume for contradiction that t₃ ⪯ t₁. Hence, t₃ \( t₁ \notin 1 \), so a \( \in t₃ \cap t₁ \) for some place a. Since t₁ takes place in \( ω₁ \) there is a token in a. If that token stays there while computation progresses in \( ω₂ \) towards t₃, then t₃ will place another token in a: contradiction with the 1-safe property. So there is a transition t₄ between t₂ and t₃ that consumes the token from a. Hence, t₄ and t₁ are in an immediate conflict and form a forward trace t₄; \dots ; t₃; t₁: contradiction.

Since t₃ and t₁ are concurrent we can apply the rule t₃; t₁ ≍ t₁; t₃, which ‘moves’ t₁ closer to t₂. We then continue this way until t₂ and t₁ are adjacent and are shown to be concurrent. Finally, we swap them and, since we get a later pair of transitions that do not agree, we conclude by induction.

(3) t₁ = t₁ and t₂ = t₂. We have assumed that t₁ \( t₂ \). Moreover, the two transitions consume different tokens, that is \( t₁ \cap t₂ = \emptyset \), otherwise we would have that t₁ \( t₂ = \emptyset \), which is impossible since occurrence nets have no backward conflicts. Since the two traces are coinitial and cofinal, then either there exists t₁ \( t₂ \in ω₁ \) or the effect of t₁ is undone later in the trace \( ω₁ \) by t₁. We now consider the two cases in turn:

(a) We can decompose the two traces as

\[
ω₁ = t₁; ω₁; ω₁ \\
ω₂ = t₂; ω₂; t₁; ω₂
\]

where \( ω₂₁ \), \( ω₂₂ \) are (potentially empty) reverse traces and \( ω₁ \) and \( ω₂ \) are forward traces. Assume that t₃ is the last transition in \( ω₂₁ \), i.e., just before t₃. Next, we show that t₃ and t₁ are concurrent, which requires t₃ ⪯ t₁ and t₁ ⪯ t₃ by Lemma 3.4 and Lemma 3.3. Since t₃; t₁ is a trace, the transitions cannot be in an immediate conflict: \( t₃ \cap t₁ = \emptyset \). This is equivalent to \( t₃ \cap t₁ = \emptyset \), which is t₁ ⪯ t₃. So, we only need to show t₁ ⪯ t₃. Assume for contradiction that t₁ ⪯ t₃. Hence, t₁ \( t₃ \notin 1 \), so a \( \in t₁ \cap t₃ \) for some place a. Since t₁ takes place in \( ω₁ \) there is a token in a. If that token stays there while computation progresses in \( ω₂ \) towards t₃, then t₃ will place another token in a: contradiction. So there is a transition t₄ between t₂ and t₃ that consumes the token from a. Hence, a \( \in t₄ \), which is a \( \in t₄ \). Combined with a \( \in t₁ \) the last given backward conflict: contradiction.

Since t₃ and t₁ are concurrent we can move t₁ closer to t₂ by using the rules of \( \approx \). We then continue this way until t₂ and t₁ are adjacent and are shown concurrent. Finally, we swap them and, since we get a later pair of transitions that do not agree, we conclude by induction.

(b) We have that \( ω₁ = t₁; ω₁; ω₁ \), where \( ω₁ \) is forward only trace that contains t₁. We show that t₁ is concurrent with all the reverse transitions in \( ω₁ \) in the corresponding way as in part (a) above. We then swap t₁ with all the reverse transitions that follow it, thus making it the last such transition of the trace. Next, we show that t₁ is concurrent with all transitions in \( ω₁ \) that appear before t₁. This is done very much as in the proof of Shortening Lemma. This allows us to swap t₁.
with all forward transitions preceding it, resulting in the sub-trace \( \hat{t}_1; t_1 \). We can then apply the axiom \( \hat{t}_1; t_1 \preceq \epsilon_m \) with \( m \) being the marking of the net before firing \( t_1 \). This gives a shorter trace equivalent to \( \omega_1 \), and we finish by induction.

With Theorem 4.6 we proved that the notion of causal consistency characterises a space for admissible rollbacks which are: (i) consistent in the sense that they do not lead to previously unreachable configurations, and (ii) flexible enough to allow rearranging of independent undo events. This implies that starting from an initial marking, all the markings reached by mixed computations are markings that could be reached by performing only forward computations. Hence, we have:

**Theorem 4.7.** Let \( O \) be an occurrence net and \( m_0 \) an initial marking. Then,

\[
(\overline{\sigma}, m_0) \rightarrow^* (\overline{\sigma}, m) \iff (\overline{\sigma}, m_0) \rightarrow^* (\overline{\sigma}, m).
\]

**Proof.** The \( \iff \) part follows trivially since \( \rightarrow \subset \rightarrow \). For the \( \Rightarrow \) part, we have that \( (\overline{\sigma}, m_0) \overset{\omega}{\rightarrow} (\overline{\sigma}, m) \) for some \( \omega \). By using Lemma 4.4, we obtain \( \omega \times \omega_1; \omega_2 \) for some forward traces \( \omega_1, \omega_2 \). Since \( \omega \times \omega_1; \omega_2 \), Theorem 4.6 gives us \( (\overline{\sigma}, m_0) \overset{\omega_1, \omega_2}{\rightarrow^*} (\overline{\sigma}, m) \). Let us note that \( m_0 \) is the initial marking, and this implies that no backward computations can take place from \( (\overline{\sigma}, m_0) \). This means that \( \omega_1 = \epsilon_m \). Hence, we have \( \omega \times \omega_2 \), where \( \omega_2 \) is a forward trace. Finally, Theorem 4.6 gives us \( (O, m_0) \rightarrow (O, m) \), which implies \( (\overline{\sigma}, m_0) \overset{\omega_2}{\rightarrow} (\overline{\sigma}, m) \) as required.

\[\Box\]

## 5. Reversing p/T nets

This section takes advantage of the classical unfolding construction for p/T nets and the reversible semantics of occurrence nets to add causally-consistent reversibility to p/T nets.

**Definition 5.1.** Let \( (N, m) \) be a marked p/T net and \( U[N, m] \) its unfolding. The reversible version of \( (N, m) \), written \( (\overline{N}, m) \), is \( \overline{U[N, m]} \).

The following result states that a reversible net is a conservative extension of its original version, namely reversibility does not change the set of reachable markings. The result is a direct consequence of Lemma 3.5 and the fact that morphisms preserve reductions [Win86, Theorem 3.1.5].

**Lemma 5.2.** \( (N, m) \rightarrow^* (N, m') \) if and only if \( (\overline{N}, m) \rightarrow^* (\overline{O}, m'') \) and \( m' = f_s(m'') \), where \( (f_s, f_t) : U[N, m] \rightarrow N \), defined such that \( f_s(a, \ldots) = a \) and \( f_t(t, \ldots) = t \), is the folding morphism.

**Proof.** Let \( U[N, m] = O \) and \( m_0 \) be such that \( m = f_s(m_0) \); namely \( m_0 \) is the set of the minimal places of \( O \). By the unfolding construction, \( (N, m) \rightarrow^* (N, m') \) implies \( (O, m_0) \rightarrow^* (O, m'') \) for an appropriate \( m'' \). Since morphisms preserve reductions [Win86, Theorem 3.1.5], \( (O, m_0) \rightarrow^* (O, m'') \) implies \( (N, m) \rightarrow^* (N, m') \). Hence, \( (N, m) \rightarrow^* (N, m') \) if and only if \( (O, m_0) \rightarrow^* (O, m'') \). Finally, we observe that \( (O, m) \uparrow (O, m') \) if and only if \( (\overline{\sigma}, m) \rightarrow^* (\overline{\sigma}, m') \) by Lemma 3.5.

\[\Box\]
We remark that the reversible version of a P/T net is defined as the reversible version of an occurrence net, namely its unfolding. Consequently, all properties shown in the previous section apply to the reversible semantics of P/T nets. In particular, Lemma 5.2 combined with Theorem 4.7 ensures that all markings reachable by the reversible semantics are just the reachable markings of the original P/T net. Formally:

**Theorem 5.3.** \((N, m) \rightarrow^∗ (N, m')\) if and only if \((\widehat N, m) \rightarrow^∗ (\widehat O, m'')\) and \(m' = f_s(m'')\), where \((f_s, f_t) : U[N, m] \rightarrow N\), defined such that \(f_S(a, \_\_\_) = a\) and \(f_T(t, \_\_\_) = t\), is the folding morphism.

**Proof.** The \(\Rightarrow\) part holds thanks to Lemma 5.2 and because \(\Rightarrow \subset \rightarrow\). For the \(\Leftarrow\) part we proceed as follows. Since \(N\) is the product of the unfolding, we have that \((\widehat N, m) \rightarrow^∗ (\widehat O, m'')\) implies \((\widehat N, m) \rightarrow^∗ (\widehat O, m'')\) by Theorem 4.7. Then we apply Lemma 5.2 and obtain \((N, m) \rightarrow^∗ (N, m')\) as required.

---

### 6. Finite Representation of Reversible P/T Nets

As seen in Figure 6(d), the reversible version of a finite net may be infinite. In this section we show how to represent reversible nets in a compact, finite way by using coloured Petri nets. We assume infinite sets \(\mathcal{X}\) of variables and \(\mathcal{C}\) of colours, defined such that \(\mathcal{X} \subset \mathcal{C}\). We use \(x, y, \ldots\) to range over \(\mathcal{X}\), \(c, d, \ldots\) to range over \(\mathcal{C}\) and \(c, d, \ldots\) to range over \(\mathcal{C} \setminus \mathcal{X}\). For \(c \in \mathcal{C}\), we write \(\text{vars}(c)\) for the set of variables in \(c\). With abuse of notation we write \(\text{vars}(m)\) for the set of variables in a multiset \(m \in \mathbb{N}^{\mathcal{P} \times \mathcal{C}}\). Let \(\sigma : \mathcal{X} \rightarrow \mathcal{C}\) be a partial function and \(c\) a colour (also, \(m \in \mathbb{N}^{\mathcal{P} \times \mathcal{C}}\)), we write \(c\sigma\) (respectively \(m\sigma\)) for the simultaneous substitution of each variable \(x\) in \(c\) (respectively \(m\)) by \(\sigma(x)\).

**Definition 6.1 (C-P/T net).** A **coloured place/transition net** (C-P/T net) is a 4-tuple \(N = (S_N, T_N, \cdot^*_N, \cdot^*_N)\), where \(S_N \subseteq \mathcal{P}\) is the (nonempty) set of places, \(T_N \subseteq \mathcal{T}\) is the set of transitions, and the functions \(\cdot^*_N, \cdot^*_N : T_N \rightarrow \mathbb{N}^{S_N \times \mathcal{C}}\) assign source and target to each transition defined such that \(\text{vars}(\cdot^*_t) \subseteq \text{vars}(\cdot^*_t)\). A marking of a C-P/T net \(N\) is a multiset over \(S_N \times \mathcal{C}\) that does not contain variables, i.e., \(m \in \mathbb{N}^{S_N \times \mathcal{C}}\) and \(\text{vars}(m) = \emptyset\). A marked C-P/T net is a pair \((N, m)\) where \(N\) is a P/T net and \(m\) is a marking of \(N\).

C-P/T nets generalise P/T nets by extending markings to multisets of coloured tokens, and transitions to patterns that need to be instantiated with appropriate colours for firing, as formally stated by the firing rule below.

\[
\frac{t = m \mid m' \in T_N}{(N, m\sigma \oplus m'') \xrightarrow{t} (N, m\sigma \oplus m'')}
\]

The firing of a transition \(t = m \mid m'\) requires to instantiate \(m\) and \(m'\) by substituting variables by colours, namely the firing of \(t\) consumes the instance \(m\sigma\) of the preset \(m\) and produces the instance \(m'\sigma\) of the postset \(m'\). Note that the initial marking of a net does not contain variables by Definition 6.1. Hence, \(m\sigma \oplus m''\) does not contain any variable. Consequently, \(\sigma(\text{vars}(\cdot^*_t)) \cap \mathcal{X} = \emptyset\). Moreover, Definition 6.1 also requires \(\text{vars}(\cdot^*_t) \subseteq \text{vars}(\cdot^*_t)\). Hence, \(m'\sigma \oplus m''\) does not contain any variable.
Example 6.2. Consider the simple c-P/T net depicted in Figure 7(a), which consists of the three places a, b and c and the coloured transition \( t = a(x) \oplus b(y) \mid c(x, y) \). The firing of \( t \) consumes a token from a and another one from b and produces a token in c, which is coloured by a pair containing the colours of the consumed tokens. Take the marking \( m = a(c_1) \oplus a(c_2) \oplus b(c_3) \). There are two possible firings of \( t \) in \( m \): one instantiates \( x \) by \( c_1 \) and \( y \) by \( c_3 \), and the other instantiates \( x \) by \( c_2 \) and \( y \) by \( c_3 \). In the former case \( m \overset{t_1}{\rightarrow} a(c_2) \oplus c(c_1, c_3) \); in the latter \( m \overset{t_2}{\rightarrow} a(c_1) \oplus c(c_2, c_3) \).

The enabling of a coloured transition may depend on the colour of the tokens. For instance, the transition \( t_1 \) in Figure 7(b) can be fired only when \( a \) contains a token coloured by a pair whose first component is \( c_1 \). Similarly, \( t_2 \) can be fired with a token from \( a \) only when its colour is a pair whose second component is \( c_2 \). In particular, the marking \( m = a(c_1, c_2) \) enables both \( t_1 \) and \( t_2 \): in the former case \( x \) is substituted by \( c_2 \) and \( m \overset{t_1}{\rightarrow} b(c_2, c_3) \); in the latter \( x \) is substituted by \( c_1 \) and \( m \overset{t_2}{\rightarrow} c(c_4) \). On the contrary, \( t_2 \) is not enabled in \( m' = a(c_1, d) \) when \( d \neq c_2 \) because there available token \( a(c_1, d) \) is not an instance of the preset of \( t_2 \), namely there is no substitution \( \sigma \) such that \( a(c_1, d) = a(x, c_2) \sigma \).

We now exploit colours to attach to each token its execution history and propose an encoding that associates each P/T net \( N \) with a C-P/T net \([N]\) whose tokens carry their execution history. Our construction resembles the unfolding constructions of P/T nets [NPW81]. Instead of using different places for representing tokens with different causal histories and different transitions for representing firings associated with different tokens, we use colour to distinguish tokens and pattern matching in transitions to distinguish firings.

Our construction relies on the set of colours \( C \) defined as the least set that contains \( \mathcal{X} \) and it is closed under the following rules.

\[
\begin{align*}
\text{(token)} & \quad h \in 2^C, \ n \in \mathbb{N} \quad \Rightarrow \quad (h, n) \in C \\
\text{(elem)} & \quad x \in \mathcal{T} \cup \mathcal{P}, \ h \in 2^C \quad \Rightarrow \quad x(h) \in C
\end{align*}
\]

Colours resemble the unfolding construction in Figure 3: the colours for tokens are \((h, n)\), where \( h \) denotes its (possibly empty) set of causes and \( n \) is a natural number used for distinguishing tokens with identical causal history. Causal histories are built up from coloured versions of transitions \((t(h))\) and places \((a(h))\).

Definition 6.3 (P/T as C-P/T). Let \( N = (S_N, T_N, \cdot_S^N, \cdot_T^N) \) be a P/T net. Then, \([N]\) is the C-P/T net defined as \([N] = (S_N, T_N, \cdot_{\circ}^{[N]}, \cdot_{\circ}^{[N]}), \) where \( 5 \)
A marked net \((N, a \oplus a)\)

In this case, from both places \(b\) generates a token in \(a\) on \(J\). The encoding of the nets in Figure 2 are shown in Figure 9. We comment Example 6.5.

These numbers are inherited by the two tokens generated by \(t\). Tokens in the initial marking have different natural numbers as second components, and in place \(a\) assigns two tokens to place \(a\) and each firing of its transition \(t_2\) produces two token in place \(b\). Its encoding as a coloured \(C\)-P/T net is shown in Figure 8(b). The colours of tokens in the initial marking have different natural numbers as second components, and these numbers are inherited by the two tokens generated by \(t_1\).

Example 6.4. Consider the marked P/T net \((N, a \oplus a)\) in Figure 8(a); its initial marking \(a \oplus a\) assigns two tokens to place \(a\) and each firing of its transition \(t_2\) produces two token in place \(b\). Its encoding as a coloured \(C\)-P/T net is shown in Figure 8(b). The colours of tokens in the initial marking have different natural numbers as second components, and these numbers are inherited by the two tokens generated by \(t_1\).

Example 6.5. The encoding of the nets in Figure 2 are shown in Figure 9. We comment on \([N_1]\), which is the encoding of \(N_1\). The transition \(t_1 = a\{c\}c\) in \(N_1\) is encoded as \(a(x)\{c(t_1(a(x)), 1)\}\), i.e., the firing of \(t_1\) consumes a token with colour \(h\) from place \(a\) and generates a token in \(c\) with colour \((t_1(a(h)), 1)\). The transition \(t_2 = b\oplus c\{e\}e\) has two places in the preset and uses two variables \(x\) and \(y\) in its encoded form \(b(x)\oplus c(y)\{e(t_2(b(x)\oplus c(y)), 1)\}\). Note that the colour of the token produced in \(c\) carries the information of the tokens consumed from both places \(b\) and \(c\). The encoding for \(t_3\) is defined analogously.

We illustrate a sequence of firings of \([N_1, a \oplus b \oplus c \oplus d]\). We remark that all the three transitions of the net \([N_1]\) are enabled at the initial marking. Let us consider the case in which \(t_1\) is fired by consuming the unique token in \(a\): \n
\[
[N_1, a \oplus b \oplus c \oplus d] = (N_1) \cdot a(0, 1) \oplus b(0, 1) \oplus c(0, 1) \oplus d(0, 1)
\]

\[
\xrightarrow{t_1}(N_1) \cdot b(0, 1) \oplus c(t_1(a(0, 1)), 1) \oplus d(0, 1)
\]

In this case, \(t_1\) consumes a token of colour \((0, 1)\) from \(a\) and produces a token of colour \((t_1(a(0, 1)), 1)\) in \(c\). The causal history of the produced token, i.e., the first component
of its colour, \( t_1(a(\emptyset,1)) \) indicates that the token has been produced by the firing of \( t_1 \) that consumed a token of colour \((\emptyset,1)\) from \( a \). In the obtained marking both \( t_2 \) and \( t_3 \) are enabled. Moreover, each transition can be fired in two different ways depending on which one of the two available tokens in \( c \) is consumed. One possible firing of \( t_2 \) is

\[
([N_1], b(\emptyset,1) \oplus c(t_1(a(\emptyset,1)),1) \oplus c(\emptyset,1) \oplus d(\emptyset,1))
\]

\[
\xrightarrow{t_2} ([N_1], e(t_2(b(\emptyset,1) \oplus c(\emptyset,1),1)) \oplus c(t_1(a(\emptyset,1)),1) \oplus d(\emptyset,1))
\]

which consumes the tokens in \( b \) and \( c \) available in the original marking. Alternatively,

\[
([N_1], b(\emptyset,1) \oplus c(t_1(a(\emptyset,1)),1) \oplus c(\emptyset,1) \oplus d(\emptyset,1))
\]

\[
\xrightarrow{t_2} ([N_1], e(t_2(b(\emptyset,1) \oplus c(t_1(a(\emptyset,1)),1)),1) \oplus c(\emptyset,1) \oplus d(\emptyset,1))
\]

where the firing of \( t_2 \) consumes the token in \( b \) generated by the previous firing of \( t_1 \).

We remark that nets in Figure 9 use colours whose second component is 1. We recall that natural numbers play the same rôle here as in the standard definition of the unfolding, which is to distinguish tokens that have the same causal history (see Section 2.2). All nets in Figure 2 have sets as initial markings and transitions that do not generate multiple tokens in the same place; consequently, the usage of natural numbers in these cases is inessential.

Although the second component in colours plays no role when reversing nets, they allow us to state a tight correspondence between the semantics of the coloured version of a \( P/T \) net and its unfolding.

**Lemma 6.6.** Let \((N,m)\) be a marked \( P/T \) net and \( U[N,m] = (O,m') \) its unfolding. Then, \([N,m] \xrightarrow{s} [N,m'') \) if and only if \((O,m') \xrightarrow{s'} (O,m'').\)

**Proof.** The \( \Leftarrow \) part is by induction on the length of the reduction. The base case follows by taking \( m'' = m' \) and noting that \([N,m] = ([N],m')\). The inductive step \( s = s' ; t \) follows by applying inductive hypothesis on \( s' \) to conclude that \([N,m] \xrightarrow{s'} ([N],m'') \) iff \((O,m) \xrightarrow{s'} (O,m'')\). If \((N,m,m''') \xrightarrow{s'} ([N],m'') \) implies \( m''' = \ast t_{[N]} + m'' \) and \( m'' = \ast t_{[N]} + m''' \). Since \((O,m) \xrightarrow{s'} (O,m'')\), \( CO(t) \). By the unfolding construction we conclude \((O,m'') \xrightarrow{s'} (O,m'')\). The \( \Rightarrow \) part follows analogously.

We now recast the notion of reversible \( P/T \) net by taking advantage of the above correspondence. In particular, we show that the reversible version of \([N]\) can be defined analogously to the case of occurrence nets, i.e., by adding transitions that are the swapped versions of the ones in \( N \).

**Definition 6.7** (Reversible \( P/T \) net). Let \( N \) be a \( P/T \) net. The reversible version of \( N \) is \( \hat{N} \). The reversible version of a marked \( P/T \) net \((N,m)\) is the marked \( C-P/T \) net \((\hat{N},[N],m)\).

We remark that \( \hat{N} \) is a \( C-P/T \) net. Note that the definition of \( C-P/T \) net (Definition 6.1) imposes \( vars(t^*) \subseteq vars(t^*) \) for any transition \( t \). Consequently, the addition of reverse transitions to a \( C-P/T \) net may not produce a \( C-P/T \) net. However, the definition of \([N]\) (Definition 6.3) and the fact that we consider nets without transitions with empty postsets ensures \( vars(t^*) = vars(t^*) \) for any \( t \in [N] \).

**Example 6.8.** Consider the \( P/T \) net \((N_2,a \oplus b \oplus c)\) in Figure 2(c), whose reversible version is shown in Figure 10(c). Such net is obtained by (i) mapping \((N_2,a \oplus b \oplus c)\) into the \( C-P/T \) net \([N_2,a \oplus b \oplus c]\) shown in Figure 9(c), and (ii) adding the reverse transitions. We now
As expected, the firing of \( t_3 \) returns because the token in \( c \) can be fired as follows:

\[
\text{(a) } [O_1]
\]

\[
\text{(b) } [(N_1, a \oplus b \oplus c)]
\]

\[
\text{(c) } [(N_2, a \oplus b \oplus c)]
\]

\[
\text{(d) } [(N_3, a)]
\]

\[
\text{(e) } [N_4]
\]

Figure 9: P/T nets as C-P/T nets

illustrate the execution of \( ([N_2], a(\emptyset, 1) \oplus b(\emptyset, 1) \oplus c(\emptyset, 1)) \). Consider the following forward computation consisting of the firing of \( t_1 \) followed by \( t_3 \).

\[
\begin{align*}
&([N_2], a(\emptyset, 1) \oplus b(\emptyset, 1) \oplus c(\emptyset, 1)) \\
&\xrightarrow{t_1}([N_2], b(\emptyset, 1) \oplus c(\emptyset, 1) \oplus d(t_3(a(\emptyset, 1), 1))) \\
&\xrightarrow{t_3}([N_2], b(\emptyset, 1) \oplus e(t_3(c(\emptyset, 1) \oplus d(t_3(a(\emptyset, 1), 1), 1), 1)))
\end{align*}
\]

Suppose, we would like to undo the above computation. The last marking only enables \( t_3 \) which can be fired as follows:

\[
\begin{align*}
&([N_2], b(\emptyset, 1) \oplus e(t_3(c(\emptyset, 1) \oplus d(t_3(a(\emptyset, 1), 1), 1), 1), 1))) \\
&\xrightarrow{t_1}([N_2], b(\emptyset, 1) \oplus c(\emptyset, 1) \oplus d(t_2(a(\emptyset, 1), 1), 1)))
\end{align*}
\]

Note that the obtained marking enables, as expected, \( t_3 \) again. Additionally, it also enables \( t_1 \) because the token in \( d \) matches the preset of \( t_1 \). As expected, the the firing of \( t_1 \) returns
Figure 10: Reversible coloured nets
to the initial marking
\[
\left(\left\lfloor N_2 \right\rfloor, b(\emptyset, 1) \oplus c(\emptyset, 1) \oplus d(t_1(a(\emptyset, 1)), 1)\right) \xrightarrow{t_2} \left(\left\lfloor N_2 \right\rfloor, a(\emptyset, 1) \oplus b(\emptyset, 1) \oplus c(\emptyset, 1)\right).
\]

Finally, we remark that the colour in the preset of the firing of \(t_2\) prevents the firing of \(t_2\) in the marking of \(\left(\left\lfloor N_2 \right\rfloor, b(\emptyset, 1) \oplus c(\emptyset, 1) \oplus d(t_1(a(\emptyset, 1)), 1)\right)\), because the colour \(t_1(a(\emptyset, 1))\) in the token in \(d\) does not match the one in the preset.

**Example 6.9.** We now illustrate the behaviour of \(\left(\left\lfloor N_3 \right\rfloor, a\right)\) in Figure 10(d) corresponding to the reversible version of \((N_3, a)\) in Figure 2(d). Consider the following forward computation consisting of two consecutive firings of \(t_1\).

\[
\begin{align*}
\left(\left\lfloor N_3 \right\rfloor, a(\emptyset, 1)\right) & \xrightarrow{t_1} \left(\left\lfloor N_3 \right\rfloor, a(t_1(a(\emptyset, 1)), 1) \oplus b(t_1(a(\emptyset, 1)), 1)\right) \\
& \xrightarrow{t_1} \left(\left\lfloor N_3 \right\rfloor, a(t_1(a(t_1(a(\emptyset, 1)), 1)), 1) \oplus b(t_1(a(t_1(a(\emptyset, 1)), 1)), 1) \oplus b(t_1(a(\emptyset, 1)), 1)\right)
\end{align*}
\]

Note that \(t_1\) can be fired infinitely many times. At any point, the last firing can be undone by applying \(t_1\). Also note that the colours in the preset of \(t_1\) ensure that the last produced token in \(b\), whose colour matches the one in \(a\), is consumed. Then, we may have the following backward computation.

\[
\begin{align*}
\left(\left\lfloor N_3 \right\rfloor, a(t_1(a(t_1(a(\emptyset, 1)), 1)), 1) \oplus b(t_1(a(t_1(a(\emptyset, 1)), 1)), 1) \oplus b(t_1(a(\emptyset, 1)), 1)\right) & \xrightarrow{t_1} \left(\left\lfloor N_3 \right\rfloor, a(t_1(a(\emptyset, 1)), 1) \oplus b(t_1(a(\emptyset, 1)), 1)\right) \\
& \xrightarrow{t_1} \left(\left\lfloor N_3 \right\rfloor, a(\emptyset, 1)\right)
\end{align*}
\]

The following result states that the reductions of the reversible c-p/t version of a net are in one-to-one correspondence with the reductions of its reversible version of its unfolding.

**Theorem 6.10 (Correctness).** Let \((N, m)\) be a marked \(p/t\) net and \(U[N, m] = (O, m')\) its unfolding. Then, \((\left\lfloor N \right\rfloor, [m]) \xrightarrow{\alpha} (\left\lfloor N \right\rfloor, [m'])\) if and only if \((\left\lfloor O \right\rfloor, [m']) \xrightarrow{\beta} (\left\lfloor O \right\rfloor, [m''])\).

**Proof.** It follows by induction on the length of the reduction analogously to the proof of Lemma 6.6. \(\square\)

7. **Conclusions**

This paper addresses the problem of reversing computations in Petri nets. The problem of reversing computation is different from the classical property of reversibility in Petri nets [CLM76], which refers to the ability of returning to its initial marking from any reachable marking without requiring to traverse back the states of the system. In this sense, making a net reversible equates to adding a minimal number of (forward) transitions that return us to the initial marking [BKMP18, ML19]. Reversibility is a global property while reversing a computation is a local one, as discussed in [BKMP18, ML19].

Reversing computation in Petri nets has also been studied in [PP18], where reversible Petri nets (RPNs) are introduced. RPNs are Petri nets endowed with a new kind of tokens, called bonds, and a computational history that records the order of execution. The motivation for bonds comes from bonds in biochemical reactions. Transitions produce bonds that are constructed from the consumed bonds. For example, the bond generated by a transition that consumes a bond from place \(a\) and bond from place \(b\) may have the shape...
a − b. In this way, each bond keeps track of its causal history and uses this information for reversal. Additionally, RPNs record the execution history in order to handle the reversal of non-deterministic choices. Informally, each firing in a computation is assigned with an integer value, which induces an ordering on firings. The main motivation to have bonds is the possibility to model out-of-causal order reversibility [PUY13, KU16, KU18]. Moreover, they are able to model also causal-consistent reversibility [DK04, PU07b, LMT14] and backtracking [HU19]. Also, a translation of RPNs into coloured Petri nets has been given [BGM⁺18]. Reversibility in RPN can be controlled by adding conditions on backward transitions. In this way it is possible to model wireless communications scenarios [PPv19].

We have presented a causally reversible semantics for P/T nets based on two observations. First, an occurrence net can be straightforwardly reversed by adding a reverse version for each (forward) transition. Second, the standard unfolding construction associates a P/T net with an occurrence net that preserves all of its computation. Consequently, the reversible semantics of a P/T net can be obtained as the reversible semantics of its unfolding. We have showed that reversibility in reversible occurrence net is causal-consistent, namely that it preserves causality. The unfolding of an occurrence net can be infinite, for example when the original P/T net is not acyclic. Therefore we have shown that the reversible behaviour of reversible occurrence nets can be expressed as a finite net whose tokens are coloured by causal histories. Colours in our encoding resemble the causal memories that are typical in reversible process calculi [LMS16, DK04]. We plan to implement an algorithm to automate our translation into coloured Petri nets and to integrate it with the CPN tool [JKW07]. This will allow us to simulate our reversible nets and eventually could provide visual analysis support for reversible debuggers [GLM14, LNPV18, HU19].

Occurrence nets have a direct mapping into prime event structures. Following the research line undertaken in [MMP⁺20], we will continue investigating in the future the relation between reversible event structures [PU15, CKV15, UPY18, GPY18] and our reversible occurrence nets. There are alternative methods for proving causal-consistent reversibility in a reversible model of computation. They are based on showing other properties than those in Section 4, either the Well-Foundedness (lack of infinite reverse sequences) and Reverse Diamond properties in [PU07b, PU07a] or the properties in [LPU20]. It would be worthwhile to prove the alternative properties for our reversible nets, and compare the two approaches.

ACKNOWLEDGMENT

We would like to thank the anonymous reviewers for their helpful suggestions of improvements.

REFERENCES


