Surgical Process Mining with Test and Flip Net Synthesis

Benoît Caillaud

Surgical Process Mining with Test and Flip Net Synthesis*

Benoît Caillaud

INRIA, Campus de Beaulieu, F-35042 Rennes cedex, France
Benoit.Caillaud@inria.fr

Abstract. Surgical process modeling aims at providing an explicit representation of surgical procedural knowledge. *Surgical process models* are inferred from a set of surgical procedure recordings, and represent in a concise manner concurrency, causality and conflict relations between actions. The paper presents preliminary results regarding the use of *test and flip* nets, a mild extension of flip-flop nets, to represent surgical process models. A test and flip net synthesis algorithm, based on linear algebraic methods in the $\mathbb{Z}/2\mathbb{Z}$ ring is detailed. Experimental results regarding the use of this synthesis algorithm to automate the construction of simple surgical process models are also presented.

1 Introduction

Process mining [1] is a very active topic and researchers have investigated methods to infer process models from recordings of actual processes for several applications: public administration workflows, manufacturing, healthcare, trafic in buildings, and recently, surgery. Surgeons pay more and more attention to the modeling of surgical procedures. The objectives are their standardization, the training of personnel (surgeons and nurses), and the development of decision assistance tools.

Contrarily to process mining methods that can rely on very large sets of recordings (database logs, ...), surgical process models must be inferred from small sets of manually recorded procedures. The consequence is that process models must generalize recordings, and retain only those meaningful causality, conflict and concurrency relations.

Test & Flip (TF) nets, a mild extension of Elementary Net Systems (with or without inhibitor arcs) [3,9] and Flip-Flop nets [10], are proposed to express surgical process models (Section 3). They offer a compromise between expressiveness and computational complexity of their synthesis (Section 6). A TF net

* Work in progress paper. This work is part of the S3PM project and has received a French government support granted to the CominLabs excellence laboratory and managed by the National Research Agency in the "Investing for the Future" program under reference ANR-10-LABX-07-01. I am indebt to Philippe Darondeau with whom I had several discussions regarding test and flip net synthesis, not long before he deceased. I would like to thank all the partners of the S3PM project, whose comments and expertise in the application field have been invaluable.

synthesis algorithm is presented (Section 5), using region theory and linear algebra in the $\mathbb{Z}/2\mathbb{Z}$ ring (Section 4).

2 Modeling Surgical Procedures

Surgical process modeling aims at providing an explicit representation of surgical procedural knowledge [8]. A *surgical process model* (SPM) represents in a concise manner concurrency, causality and conflict relations between actions, that can be inferred from a set of *surgical procedure recordings* (SPR). These SPRs are usually acquired manually with a tablet computer, during actual surgical operations[1]. The accurate identification of each action of the personnel (surgeons, nurses, anethtetist) requires expert knowledge and the acquisition is usually performed by a surgeon.

A SPM is built from several (tens to hundreds of) SPR of the same type of operation into a single comprehensive model. The model is expected to be used in the following contexts:

- standardization of surgical practice,
- teaching and assessment of surgical procedures,
- decision assistance during operations.

In any case, SPMs alone are not meant to be exploited directly by medical personnel, but rather to be used in simulation and analysis tools. Since recordings can not be exhaustive, process models are expected to generalize recorded procedures and allow unrecorded but meaningful scenarios. Synthesis of a SPM is therefore an iterated process where the synthesized model has to be validated by experts. Interactive simulation is expected to be the simplest way to validate a SPM. Whenever synthesis produces models with unwanted scenarios, a refinement of the ontology leads to a narrowing of the process model, with fewer generalizations.

A SPR is a finite sequence of actions which are tuples of the form *(phase, hand, tool, organ, type)*. A surgical operation is decomposed into several phases, for instance *opening, approach, resection, closing*. Actions are performed by one of the two hands of a surgeon (*left hand surgeon 1, right hand surgeon 2*), of the anethtetist, or of the nurses. Tools are any of the surgical tools used during the operation: *scalpel, scissors, suction tube, forceps*. The organ is any of the patient's organ, for instance *skin, muscle, lumbar disk*. The last component of the tuple is a type of action: *cut, hold, coagulate, install, dissect, irrigate*.

Surgical procedures have several generic properties, that should be taken in consideration for the choice of formalism used to express SPMs. Manual inspection of several SPRs and exchanges with surgeons revealed a striking feature of surgical procedures: Actions that can be repeated twice, can also be repeated an arbitrary number of times. It is not uncommon to find in procedure recordings, sequences of the form *rinse, stitch, rinse, stitch, ...*. The number of iterations

[1] `https://medicis.univ-rennes1.fr/index/software/procside`

is variable and depends on several hidden parameters that are not taken into account in the SPM. This includes the patient's morphology, and the surgeon's habit. The consequence is that counting is not required, but moreover, it should not be allowed in process models. The reason is that we want process model synthesis to generalize a set of procedure recordings and allow arbitrary iterations of repeated actions or sequences of actions, instead of putting a bound on the number of iterations. The easiest way to do this is to use a formalism where counting is not possible, for instance, a class of binary Petri nets.

Concurrent activities often appear in surgical procedures. Concurrency naturally arises from the activity of the surgeons, anethtetist and nurses. Concurrency may also be caused by the independence between two sequences of actions, that can be performed in any order, or even interleaved.

3 Test & Flip Nets

Petri nets are perfectly suited to express concurrency, causality and conflict. The main generic property of surgical procedures is that actions or sequences of actions that can be repeated twice, can also be repeated an arbitrary number of times. The consequence is that counting is not required, but moreover, it should not be allowed in process models, in order to generalize few instances of repetitions, into an arbitrarily long iteration. The easiest way to do this is to use a class of Petri nets where counting is not possible, for instance, a class of binary nets.

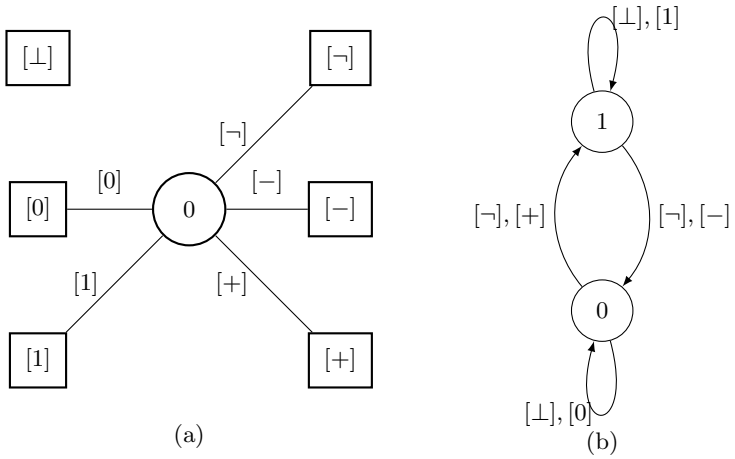

Fig. 1. (a) One place TF net and (b) its marking graph

We propose *test & flip* (TF) nets, a mild extension of *flip-flop* nets [10], *elementary* nets [3] and elementary nets with *inhibitor* arcs [9]. TF nets are

binary nets where transitions test the markings of a set of places (is the marking equal to 0 or 1?), and if the test is successful, flip (complement) the markings of another set of places. The behavior of an arbitry TF net is completely defined by the marking graph of the one-place net given Fig. 1. Remark that there are six types of flow arcs $[\bot], [0], [1], [\neg], [-], [+]$, contrarily to flip-flop nets, where flow arcs can be of types $[\bot], [\neg], [-], [+]$. Arcs of type $[\bot]$ may be implicit, meaning that if no arc is represented between a place and a transition, this implies that they are connected by a $[\bot]$ arc.

Remark that TF nets are strictly more expressive than flip-flop nets and elementary nets with inhibitor arcs. For instance, the graph given on the left-hand side of Fig. 2 is isomorphic to the marking graph of no flip-flop net, while it is isomorphic to the marking graph of the TF net shown on the right-hand side of the figure.

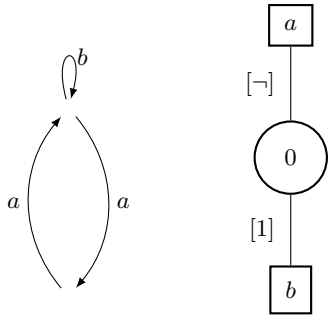

Fig. 2. (a) Graph that can not be realized by a FF net, (b) its realization as a TF net

In the sequel, linear algebraic methods are used and the following definition of TF nets simplifies notations:

Definition 1 (Test & Flip Net). *A test & flip net is a tuple* $N = (P, T, a, b, c, m_0)$ *where:*

- P *is a set of places and* T *is a set of transitions.*
- $a, b, c : P \times T \to \mathbb{Z}/2\mathbb{Z}$ *are three mappings defining the effect of a transition on the places.*
- $m_0 : P \to \mathbb{Z}/2\mathbb{Z}$ *is the initial marking.*

The behavior of a TF net is defined by the following rules:

- *A transition* $t \in T$ *is enabled in marking* $m : P \to \mathbb{Z}/2\mathbb{Z}$*, denoted* $m \xrightarrow{t}$ *, iff for all places* $p \in P$*,* $a(p,t) \cdot m(p) + b(p,t) \equiv 0$ (2).

– *The firing of transition $t \in T$ changes the marking from m to m', denoted $m \xrightarrow{t} m'$ iff $m \xrightarrow{t}$ and for all places $p \in P$, $m'(p) \equiv m(p) + c(p,t)$ (2).*

Remark that not all assignments of a, b and c are meaningful: $(a,b,c) = (0,1,0)$ and $(a,b,c) = (0,1,1)$ make the transition non fireable. Therefore, there are six valid assignments, corresponding to the six types of flow arcs, as listed in table 1.

	[⊥]	[0]	[1]	[¬]	[+]	[−]
a	0	1	1	0	1	1
b	0	0	1	0	0	1
c	0	0	0	1	1	1

Table 1. Assignments corresponding to the six types of flow arcs

Relation $\xrightarrow{t}$ is extended to sequences of transitions: for all markings m, $m \xrightarrow{\epsilon}{}^{\star} m$ holds and, for all markings m, m', $u \in T^\star$ and $t \in T$, $m \xrightarrow{u.t}{}^{\star} m'$ holds iff there exists a marking m'' such that $m \xrightarrow{u}{}^{\star} m'' \xrightarrow{t} m'$. The enabledness predicate is also extended to sequences of transitions: $m \xrightarrow{u}{}^{\star}$ holds iff there exists a marking m' such that $m \xrightarrow{u}{}^{\star} m'$.

The following notations are used in the sequel of the paper. Given two vectors $u, v : T \to \mathbb{Z}/2\mathbb{Z}$, $\langle u, v \rangle$ denotes the scalar product of the two vectors: $\langle u, v \rangle \equiv \sum_{t \in T} u(t).v(t)$ (2). Vector $\mathbf{1}_t$ is the unit vector along t: $\mathbf{1}_t(t) \equiv 1$ (2) and for every $s \neq t$, $\mathbf{1}_t(s) \equiv 0$ (2). The empty word in $T^\star$ is denoted ϵ. We now define the Parikh image modulo 2 of a word:

Definition 2 (Parikh image). *The Parikh image of sequence $u \in T^\star$, denoted $\pi(u)$, is the commutative image of u modulo 2. Hence it is a mapping $\pi : T^\star \to T \to \mathbb{Z}/2\mathbb{Z}$, such that $\pi(\epsilon) \equiv \mathbf{0}$ (2) and, $\forall u \in T^\star, t \in T, \pi(u.t) \equiv \pi(u) + \mathbf{1}_t$ (2).*

Remark that TF nets can not distinguish transition sequences with identical parikh images, if they are both enabled: for all markings m, m', m'' and for all sequences $u, v \in T^\star$ such that $m \xrightarrow{u}{}^{\star} m'$ and $m \xrightarrow{v}{}^{\star} m''$, $\pi(u) \equiv \pi(v)$ (2) implies $m' \equiv m''$ (2).

The language of a TF net $N = (P, T, a, b, c, m_0)$ is the prefix-closed language of enabled sequences of transitions: $\mathcal{L}(N) = \{u \in T^\star | m_0 \xrightarrow{u}{}^{\star}\}$.

4 TF Regions

We now use the theory of regions [4], adapted to TF nets. The definition of TF regions follows that of P/T regions, with the difference that TF regions are defined on the $\mathbb{Z}/2\mathbb{Z}$ ring, instead of $\mathbb{Z}$ or $\mathbb{Q}$.

Regions corresponding to places of a TF net are defined as follows:

Definition 3 (TF Region). *Given a transition system* $S = (T, Q, q_0 \in Q, \rightarrow \in Q \times T \times Q)$*, a TF region of* S *is a tuple* $\rho = (m : Q \rightarrow \mathbb{Z}/2\mathbb{Z}, a, b, c : T \rightarrow \mathbb{Z}/2\mathbb{Z})$ *such that for all transitions* $q \xrightarrow{t} q'$*,*

$$\begin{cases} a(t).m(q) + b(t) \equiv 0\ (2) \\ m(q') \equiv m(q) + c(t)\ (2) \end{cases}$$

Remark that when the transition system is reachable, a region is uniquely defined by its value $m(q_0)$ in the initial state and the three mappings a, b and c. Indeed, consider a spanning tree $S' \subseteq S$ rooted in q_0. Mapping m can be reconstructed by induction: $\forall q \xrightarrow{t} q' \in S'$, $m(q') \equiv m(q) + c(t)\ (2)$. In the sequel we will assume transition systems to be reachable.

Regions will be used to solve event-state speration problems, defined below.

Definition 4 (Event-State Separation Problem (ESSP)). *Given a transition system* $S = (T, Q, q_0 \in Q, \rightarrow \subseteq Q \times T \times Q)$ *and a pair* $(q, t) \in Q \times T$ *such that no transition labelled* t *exits state* q*, region* $\rho = (m : Q \rightarrow \mathbb{Z}/2\mathbb{Z}, a, b, c : T \rightarrow \mathbb{Z}/2\mathbb{Z})$ *is a solution to the event-state separation problem (ESSP)* (q, t)*, denoted* $\rho \models (q, t)$*, iff* $a(t).m(q) + b(t) \equiv 1\ (2)$*.*

By symmetry, regions can be assumed to be initialized to 0. Indeed, consider a region $\rho = (m : Q \rightarrow \mathbb{Z}/2\mathbb{Z}, a, b, c : T \rightarrow \mathbb{Z}/2\mathbb{Z})$, initialized to $m(q_0) \equiv 1\ (2)$ and solution of ESSP (q, t), meaning that $a(t).m(q) + b(t) \equiv 1\ (2)$. Complementary region $\rho' = (m' \equiv 1 + m\ (2), a' \equiv a\ (2), b' \equiv a + b\ (2), c' \equiv c\ (2))$ is a region that is also solution of ESSP (q, t).

TF regions generalize elementary regions, defined for the synthesis of elementary net systems [3]. However, unlike elementary regions, TF regions can not be uniquely reconstructed from their support m. Instead of representing a region by its support, it will be represented by the tuple (a, b, c). The support can be reconstructed, assuming that it is initialized to $m(q_0) \equiv 0\ (2)$. By abuse of notation and from now on, the support of a region will be omitted, and $\rho = (a, b, c)$ denotes the unique regions $\rho = (m, a, b, c)$ where support m is initialized to 0. Two regions $\rho = (a, b, c)$ and $\rho' = (a', b', c')$ have identical supports, $m \equiv m'\ (2)$ iff $c \equiv c'\ (2)$.

The signature of a region $\rho = (a, b, c)$ is the first component, a, of the tuple. It defines the set of transitions that are tested by the one-place net defined by the region. Given $U \subseteq T$, a region $\rho = (a, b, c)$ is said to be a U-region, meaning that its signature is equal to U, iff $\forall t, a(t) \equiv 1\ (2) \iff t \in U$. Given $t \in T$, a t-region is a region of signature $\{t\}$. Given a region $\rho = (a, b, c)$, the t-projection of region ρ is the region $\rho_t = (a_t, b_t, c_t)$, where $a_t(t) \equiv a(t)\ (2)$, $b_t(t) \equiv b(t)\ (2)$, $\forall t' \neq t$, $a_t(t') \equiv b_t(t') \equiv 0\ (2)$, and $c_t \equiv c\ (2)$.

Consider a transition system $S = (T, Q, q_0, \rightarrow)$, and a spanning tree $S' \subseteq S$, rooted in q_0. The Parikh mapping is extended to states, $\pi : Q \rightarrow T \rightarrow \mathbb{Z}/2Z$:

$$\begin{cases} \pi(q_0) \equiv \mathbf{0}\ (2) \\ \forall (q, t, q') \in S'\ \ \pi(q') \equiv \pi(q) + \mathbf{1}_t\ (2) \end{cases}$$

Given a region $\rho = (a, b, c)$, solving ESSP (q, t). Remark that the t-projection of ρ is also a solution of ESSP (q, t). Indeed, regions ρ and ρ_t have identical supports, $\forall q, m(q) \equiv m_t(q)$ (2), and $a_t(t).m_t(q)+b_t(t) \equiv a(t).m(q)+b(t) \equiv 1$ (2). This allows to solve event-state separation problems of the form (q, t) with t-regions only. Remark that t-regions are solution of the following system of linear equations:

$$\forall (q, t) \text{ st. } q \xrightarrow{t} , \ \langle \pi(q), c \rangle + b(t) \equiv 0 \ (2)$$

Therefore, the set of t-regions form a vector space on the ring $\mathbb{Z}/2\mathbb{Z}$. This allows to use linear algebraic methods to solve ESSPs and synthesize TF nets.

Given two regions ρ and ρ', ρ subsumes ρ', denoted $\rho \succeq \rho'$, iff for all ESSP (q, t), ρ' solves (q, t) implies ρ solves (q, t). Consider two disjoint sets of labels $U, U' \subseteq T$, $U \cap U' = \emptyset$, a U-region $\rho = (a, b, c)$, and a U'-region $\rho' = (a', b', c')$. Assume ρ and ρ' have identical supports, meaning that $c \equiv c'$ (2). Define region $\rho + \rho' = (a + a', b + b', c)$ to be a $U \cup U'$ region. Remark that every ESSP (q, t) solved either by ρ or by ρ' is also solved by $\rho+\rho'$. Hence $\rho+\rho' \succeq \rho, \rho'$. This allows to merge regions with identical supports into regions with larger signatures.

5 TF Net Synthesis

Like P/T net synthesis [7], linear algebraic methods are used. The main difference is that modular arithmetics is used in place of rational arithmetics. The following problem is solved:

Definition 5 (TF Net Synthesis Problem). *Given an alphabet T and a prefix-closed regular language $A \subseteq T^\star$, compute a TF net $N = (P, T, a, b, c, m_0)$ such that the language of N is the least language of TF nets containing A:*

$$\begin{cases} A \subseteq \mathcal{L}(N) \\ \forall N', \ A \subseteq \mathcal{L}(N') \Rightarrow \mathcal{L}(N) \subseteq \mathcal{L}(N') \end{cases}$$

The synthesis method consists in five phases, that can be described informally as follows:

1. Compute a quotient transition system from A, modulo Parikh image in $\mathbb{Z}/2\mathbb{Z}$.
2. Compute the sets of linear equations $\mathcal{R}_t$ defining t-regions of the above transition system, for each $t \in T$.
3. Compute a set of *admissible* t-regions, solving a maximum set of ESSPs. Whenever a separation problem has no solution, the transition system is extended with a new transition. New states may be added, introducing new separation problems.
4. Eliminate redundent regions.
5. Merge regions with identical supports and disjoint signatures.

Phases 1 and 2 is nothing more than the application of the definition of t-regions. We now detail phase 3 of the algorithm, before proving the correctness and optimality of the method:

procedure TFSynthesis($T, A \subseteq T^\star, (\mathcal{R}_t)_{t\in T}$))
  $\Pi \leftarrow \pi(A)$ ▷ States of the quotient transition system
  $\Delta \leftarrow \{(\pi(u), t) | u.t \in A\}$ ▷ Transitions of the quotient transition system
  $\Sigma \leftarrow (\Pi \times T) \setminus \Delta$ ▷ ESSP set
5: $\Omega \leftarrow \emptyset$ ▷ Set of admissible regions
  while $\Sigma \neq \emptyset$ **do** ▷ Is there an ESSP to solve?
    choose $(\pi, t) \in \Sigma$ ▷ Select an ESSP
    solve $\mathcal{R}_t \cup \{\langle \pi, c\rangle + b(t) \equiv 1\ (2)\}$ ▷ Solve ESSP
    if exists solution $\rho = (a, b, c)$ **then** ▷ Is there a solution?
10: $\Omega \leftarrow \Omega \cup \{\rho\}$ ▷ Yes: Add solution to admissible set
      $\Sigma \leftarrow \Sigma \setminus \{(\pi', t) | \rho \models (\pi', t)\}$ ▷ Remove all ESSPs solved by ρ
    else ▷ No:
      $\Sigma \leftarrow \Sigma \setminus \{(\pi, t)\}$ ▷ Remove ESSP
      $\pi' \leftarrow \pi + \mathbf{1}_t\ (2)$ ▷ State reached by new transition
15: $\Delta \leftarrow \Delta \cup \{(\pi, t)\}$ ▷ Add new transition
      if $\pi' \notin \Pi$ **then** ▷ Is state new?
        $\Pi \leftarrow \Pi \cup \{\pi'\}$ ▷ Yes: Add new state
        $\Sigma \leftarrow \Sigma \cup \{(\pi', t') | \forall \rho \in \Omega, \rho \not\models (\pi', t')\}$ ▷ Add ESSPs
      end if
20: **end if**
  end while
  return Ω ▷ Return the set of admissible regions
end procedure

Correctness of the algorithm, $A \subseteq \mathcal{L}(N)$, is a consequence of the two following properties:

- First of all, A is included in the language $\mathcal{L}(S)$ of the quotient transition system $S = (T, \Pi, \mathbf{0}, \Delta)$.
- Then, N consists in places corresponding to regions of S. Therefore, for every $u.t \in \mathcal{L}(S)$, if $u \in \mathcal{L}(N)$ then transition t is enabled in the marking m reached by firing sequence u. Since $\mathcal{L}(N)$ is prefix-closed and by induction on the length of u, $\mathcal{L}(S) \subseteq \mathcal{L}(N)$.

Optimality is more involved and uses the following properties:

- Using the fact that TF nets can not distinguish firing sequences that are Parikh equivalent modulo 2, for all TF net N, $A \subseteq \mathcal{L}(N)$ implies $\mathcal{L}(S) \subseteq \mathcal{L}(N)$. Hence, every optimal approximation of $\mathcal{L}(S)$ as a TF net language is also an optimal approximation of A.
- In the while loop, the synthesis algorithm computes an increasing sequence of transition systems $S_0 \dots S_{n+1}$, where S_i is the transition system defined by the value of (Π, Δ) when the while loop test is evaluated for the $i+1$th time (line 6). Define $(\pi_i, t_i)_{i=0\dots n}$ to be the sequence of ESSPs chosen line 7. Whenever ESSP (π_i, t_i) admits a solution, $S_{i+1} = S_i$. If (π_i, t_i) has no solution, $S_{i+1} = S_i \cup \{(\pi_i, t_i)\}$, by adding a transition (line 15) and, possibly, a state (line 17). We shall prove that for all $t \in T$, transition systems

$(S_i)_{i=0\ldots n}$ have isomorphic vector spaces of t-regions. The interesting case is when S_{i+1} is a strict extension of S_i, meaning that ESSP (π_i, t_i) admits no solution. Hence, the system of linear equations line 8 has no solution. Therefore every solution $\rho = (a, b, c)$ of $\mathcal{R}_t$ satisfies $\langle \pi, c \rangle + b(t) \not\equiv 1$ (2), or equivalently $\langle \pi, c \rangle + b(t) \equiv 0$ (2). Therefore constraint $\langle \pi, c \rangle + b(t) \equiv 0$ (2) is redundent with $\mathcal{R}_t$, and S_i and S_{i+1} have isomorphic vector spaces of t-regions.

– Remark that for all $i = 0 \ldots n$, for all $u \in \mathcal{L}(S_{i+1}) \setminus \mathcal{L}(S_i)$, and for all TF net N, $\mathcal{L}(S_i) \subseteq \mathcal{L}(N)$ implies $u \in \mathcal{L}(N)$. This implies that for all TF net N, $\mathcal{L}(S_0) \subseteq \mathcal{L}(N)$ implies $\mathcal{L}(S_{n+1}) \subseteq \mathcal{L}(N)$. Which proves optimality of the language of the TF net defined by the set of admissible regions Ω: $\mathcal{L}(N_\Omega) = \mathcal{L}(S_{n+1})$ is the least language of TF nets containing $\mathcal{L}(S)$.

Phases 4 and 5 of the synthesis method perform transformations on the set of admissible regions wihtout changing the language of the synthesized net. Phase 4 consists in discarding redundent regions, one per one, in the reverse order of their addition to set Ω. These are regions ρ such that for all ESSP (π, t) solved by ρ, there exists a distinct admissible region $\rho' \in \Omega$, $\rho' \neq \rho$ such that ρ' solves (π, t). Phase 5 consists in merging all admissible regions with identical supports.

6 Example

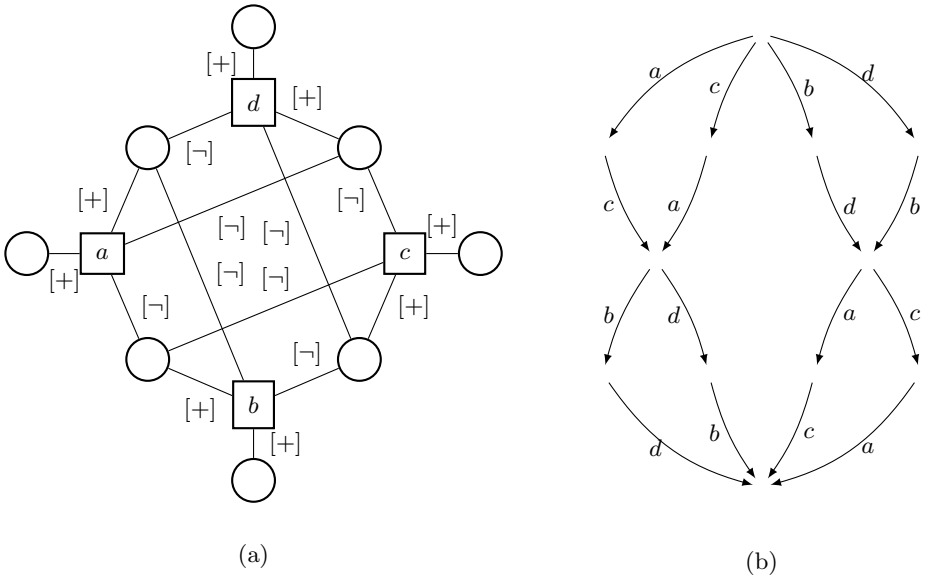

Fig. 3. (a) Synthesized TF net and (b) its marking graph

This example is inspired by a mechanical engineering procedure used whenever a set of bolts, organized in a circle, must be tightened. Informally, as soon as a bolt has been tightened, the opposite one must also be tightened, before tightening any other bolt. There are four bolts a, b, c and d. We would like to compute a model of the procedure, using a few correct instances of the informal procedure. The procedure instances are as follows:

$$A = \{ acbd, dbca, bdac \}$$

Fig. 3 shows the synthesized TF net. It generalizes the three procedure instances into a process model allowing eight possible orderings of the four actions:

$$\mathcal{L}(N) = \{ acbd, acdb, cabd, cadb, bdac, bdca, dbac, dbca \}$$

Remark that only three instances have been necessary and that the synthesized net captures the expected causality and conflict relations. This process model can not be represented as an elementary net, unless label splitting is used [5], in which case the resulting net is far more complex than the TF net.

7 Conclusion

Test & Flip (TF) nets have been introduced as models of surgical procedures. These nets are strictly more expressive than Elementary Net Systems [3], with or without inhibitor arcs [9], and Flip-Flop nets [10]. A TF net synthesis algorithm, implemented in a software prototype, has been designed, using region theory and linear algebra in the $\mathbb{Z}/2\mathbb{Z}$ ring. The TF net synthesis algorithm presented Section 5 has been implemented in a software tool[2]. This tool parses procedure recordings in the XES file format[3], a standard XML interchange format supported by several process mining tools. Linear algebra in $\mathbb{Z}/2\mathbb{Z}$ has been implemented in a C library, while the core of the synthesis algorithm, parser, and post-processing have been implemented in OCaml. Despite the synthesis algorithm has an exponential worst case time complexity, and that performances degrade steeply if concurrency in the resulting net is massive, the tool performs well on large sample XES logs. The main limitation is not the number of procedure recordings or the length of each recording, but rather the size of the alphabet of actions. However, this should not be an issue for the targeted application, where concurrency is confined within short sequences of actions. Typical surgical procedure recordings consist in 50 to 200 operations containing each about 100 to 500 actions, for a total of less than 200 action types.

Preliminary tests have been successful on simplified recordings of lumbar disc operations: 5 recordings of about 120 actions each. Qualitative analysis of the synthesized nets could not be carried out, since this requires expert knowledge in surgery. This is planned later in the course of the S3PM project[4], as soon as

[2] `http://tinyurl.com/oql6f3y`
[3] `http://www.xes-standard.org/`
[4] `http://www.cominlabs.ueb.eu/themes/project/`

a complete tool chain can be used by surgeons. This tool chain will integrate the acquisition of procedure recordings, the synthesis of process models and their simulation. Comparison with other process mining methods and tools [2] will then be possible. Research on the application of TF net synthesis to surgical process mining will continue on the following directions:

- The computation of a process model uses detailed recordings of surgical procedures and an ontology of possible actions. Whenever the inferred process model makes abusive generalizations, a possible strategy towards a more precise model is to refine the ontology. Label splitting can be used to assist the refinement of action ontologies. Label splitting heuristics have been developed for Elementary Net Systems [5,6]. The objective is to adapt these heuristics to the more expressive TF nets.
- Decomposition of a SPM into a product of sequential components. Like for elementary regions, minimal TF regions exist and a partitioning of the state space into minimal regions defines a sequential component.
- A longer term research topic is related to the fact that procedure recordings are acquired manually and may contain errors. The use of stochastic methods to filter out isolated behavior (for instance, using Bayesian networks) would help the synthesis of accurate and meaningful process models.

References

1. van der Aalst, W.M.P.: Process mining. Commun. ACM 55(8), 76–83 (2012)
2. van der Aalst, W.M.P., van Dongen, B.F., Günther, C.W., Rozinat, A., Verbeek, E., Weijters, T.: Prom: The process mining toolkit. In: de Medeiros, A.K.A., Weber, B. (eds.) BPM (Demos). CEUR Workshop Proceedings, vol. 489. CEUR-WS.org (2009)
3. Badouel, E., Bernardinello, L., Darondeau, P.: The synthesis problem for elementary net systems is np-complete. Theor. Comput. Sci. 186(1-2), 107–134 (1997)
4. Badouel, E., Darondeau, P.: Theory of regions. In: Reisig, W., Rozenberg, G. (eds.) Lectures on Petri Nets I: Basic Models, Lecture Notes in Computer Science, vol. 1491, pp. 529–586. Springer Berlin Heidelberg (1998), http://dx.doi.org/10.1007/3-540-65306-6_22
5. Carmona, J.: The label splitting problem. T. Petri Nets and Other Models of Concurrency 6, 1–23 (2012)
6. Carmona, J., Cortadella, J., Kishinevsky, M.: Genet: A tool for the synthesis and mining of petri nets. In: ACSD. pp. 181–185. IEEE Computer Society (2009)
7. Darondeau, P.: Region based synthesis of p/t-nets and its potential applications. In: ICATPN. pp. 16–23 (2000)
8. Jannin, P., Morandi, X.: Surgical models for computer-assisted neurosurgery. In: Neurimage'07 (2007)
9. Pietkiewicz-Koutny, M.: Transition systems of elementary net systems with inhibitor arcs. In: Azma, P., Balbo, G. (eds.) Application and Theory of Petri Nets 1997, Lecture Notes in Computer Science, vol. 1248, pp. 310–327. Springer Berlin Heidelberg (1997), http://dx.doi.org/10.1007/3-540-63139-9_43
10. Schmitt, V.: Flip-flop nets. In: Puech, C., Reischuk, R. (eds.) STACS. Lecture Notes in Computer Science, vol. 1046, pp. 517–528. Springer (1996)