Algorithm to generate finite negative tomonoids

Milan Petrík[1,2,⋆] and Thomas Vetterlein[3,⋆⋆]

1. Department of Mathematics, Faculty of Engineering,
Czech University of Life Sciences, Prague, Czech Republic;
2. Institute of Computer Science, Academy of Sciences, Prague, Czech Republic
petrikm@tf.czu.cz,petrik@cs.cas.cz
3. Department of Knowledge-Based Mathematical Systems,
Johannes Kepler University, Linz, Austria
Thomas.Vetterlein@jku.at

Abstract. The canonical semantics of fuzzy logic is typically based on negative totally ordered monoids. This contribution describes an algorithm generating in a step-wise fashion all finite structures of this kind.

Keywords: finite negative totally ordered monoid, Rees coextension, Rees congruence, Reidemeister closure condition, tomonoid partition

1 Introduction

Partially ordered monoids are structures occurring in several fields of mathematics and computer sciences, in particular in logic. Indeed, in non-classical logic, the canonical set of truth values is often endowed with a binary operation making this set into a partially ordered monoid. The monoidal operation then corresponds to the conjunction. Moreover, in fuzzy logic the partially ordered semantics can be represented by linear ones and often the top element is the monoidal identify. The resulting structures are negative totally ordered monoids.

In this contribution, we focus on finite structures as they may be used, e.g., in finite-valued fuzzy logics. We note that, under the additional assumption of commutativity, the structures that we consider can be identified with linearly ordered finite MTL-algebras; MTL-algebras are in turn the algebraic counterpart of the fuzzy logic MTL [3].

Further, this contribution can be seen as a practical appendix to our previous paper [9] which has yielded a method to describe all the one-element Rees coextensions of a given finite, negative, totally ordered monoid (shortly a f.n. tomonoid) S, that is, all the f.n. tomonoids greater by one element such that S is their common Rees quotient. This way, starting from the trivial monoid, one can generate all the possible f.n. tomonoids up to a given finite size. While the

⋆ M. P. was supported by the Czech Science Foundation (GAČR) under Project 15-07724Y (Totally ordered monoids).

⋆⋆ Th. V. was supported by the Austrian Science Fund (FWF) under Project I 1923-N25 (New perspectives on residuated posets).

cited paper has been focused on describing the coextensions and giving a proof that all the existing f.n. tomonoids are necessarily obtained this way, this paper intends to give a practical description of the algorithm that produces the tomonoids. This paper can be also viewed as a continuation of our previous result [7] where we have, however, dealt with the Archimedean case only.

2 Basic notions

We begin with an introduction of the basic notions of the paper. A *monoid* is an algebra $(S; \odot, 1)$ of the type $\langle 2, 0\rangle$ such that $(a \odot b) \odot c = a \odot (b \odot c)$ and $a \odot 1 = 1 \odot a = a$ for every $a, b, c \in S$. A total (linear) order $\leqslant$ on a monoid S is called *compatible* if $a \leqslant b$ implies both $a \odot c \leqslant b \odot c$ and $c \odot a \leqslant c \odot b$ for every $a, b, c \in S$. In such a case, we call $(S; \leqslant, \odot, 1)$ a *totally ordered monoid* or a *tomonoid*, for short. We also say that $\odot$ is *monotone* with respect to $\leqslant$. Further, S is called *commutative* if $a \odot b = b \odot a$ for every $a, b \in S$. Finally, S is called *negative* (also *integral*) if 1 is the top element.

This paper is focused mainly on finite, negative, totally ordered monoids which we abbreviate by "*f.n. tomonoids*". In general, we do not assume the monoids to be commutative [4], although, we deal with the commutativity, as well. The smallest monoid that consists of the monoidal identity 1 alone, is called the *trivial tomonoid*.

3 Level set representation of tomonoids

We do not work with f.n. tomonoids directly but we rather work with their level set representations. In the sequel, by S^2 we denote the cartesian product of the set S with itself, i.e., $S^2 = S \times S$.

Let $(S; \leqslant, \odot, 1)$ be a tomonoid. For two pairs $(a, b), (c, d) \in S^2$ we define

$$(a, b) \sim (c, d) \quad \text{iff} \quad a \odot b = c \odot d$$

and we call $\sim$ the *level equivalence* associated with S.

Let $(S; \leqslant)$ be a totally ordered set. By $\trianglelefteq$ we denote the componentwise order on S^2, i.e., for every $a, b, c, d \in S$, we put

$$(a, b) \trianglelefteq (c, d) \quad \text{iff} \quad a \leqslant b \quad \text{and} \quad c \leqslant d.$$

Let $1 \in S$ and let $\sim$ be an equivalence on S^2 such that the following holds:

(P1) For every $a, b, c, d, e \in S$, $(a, b) \sim (1, d)$ and $(b, c) \sim (1, e)$ imply $(d, c) \sim (a, e)$. (See an illustration in Figure 1-left.)
(P2) For every $a, b \in S$ there is exactly one $c \in S$ such that $(a, b) \sim (1, c) \sim (c, 1)$.
(P3) For every $a, b, c, d, a', b', c', d' \in S$, $(a, b) \sim (a', b') \trianglelefteq (c, d) \sim (c', d') \trianglelefteq (a, b)$ implies $(a, b) \sim (c, d)$.

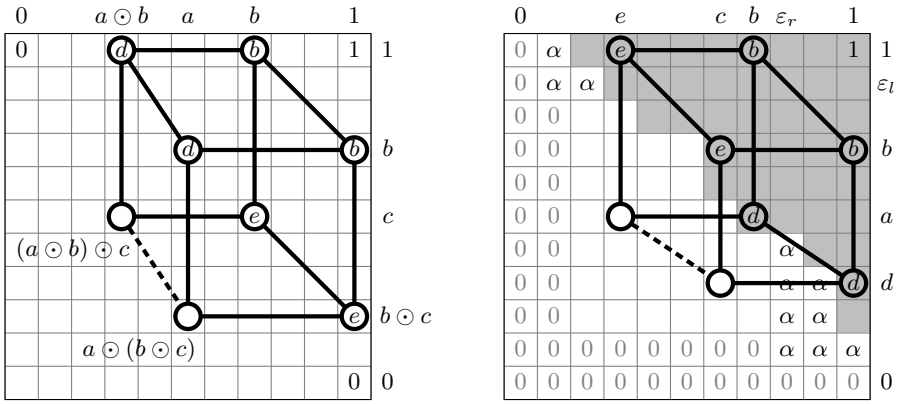

Fig. 1. Left: Illustration of Property (P1). Consider two rectangles such that the first one hits the upper edge and the second one hits the right edge. Assume that the upper left, upper right, and lower right vertices are in the same level equivalence classes, respectively. Then also the remaining lower left vertices are elements of the same level equivalence class. This property is directly related to the associativity of the tomonoid and corresponds with the *Reidemeister condition* known from web geometry [1, 2]. **Right: Illustration of (E2).** For every two pairs $(a, b), (b, c) \in \mathcal{P}$ we relate $(a, e) \dot{\sim} (d, c)$.

Then we call $(S^2; \trianglelefteqslant, \sim, (1,1))$ a *tomonoid partition*. The following two propositions show that tomonoids and tomonoid partitions are in a one-to-one correspondence.

Proposition 1. *[9] Let $(S; \leqslant, \odot, 1)$ be a tomonoid and let $\sim$ be its level equivalence. Then $(S^2; \trianglelefteqslant, \sim, (1,1))$ is a tomonoid partition.*

Proposition 2. *[9] Let $(S^2; \trianglelefteqslant, \sim, (1,1))$ be a tomonoid partition. For every $a, b \in S$, let $a \odot b$ be given as the unique c such that $(a, b) \sim (1, c) \sim (c, 1)$. Then $(S; \leqslant, \odot, 1)$ is the unique tomonoid such that $(S^2; \trianglelefteqslant, \sim, (1,1))$ is its associated tomonoid partition.*

In the sequel, we will write $(a, b) \sim c$ instead of $(a, b) \sim (1, c) \sim (c, 1)$.

4 Rees quotients and coextensions

In this section we introduce the notion of a one-element coextension of a f.n. tomonoid.

Let $(S; \leqslant, \odot, 1)$ be a f.n. tomonoid. Its least element we call the *zero* (and we denote it by 0), its second smallest element we call the *atom* (and we denote it by α), and its second greatest element we call the *coatom* (and we denote it by κ). Recall that 1 is the greatest element of S.

A *tomonoid congruence* on S is an equivalence relation $\approx$ on S such that

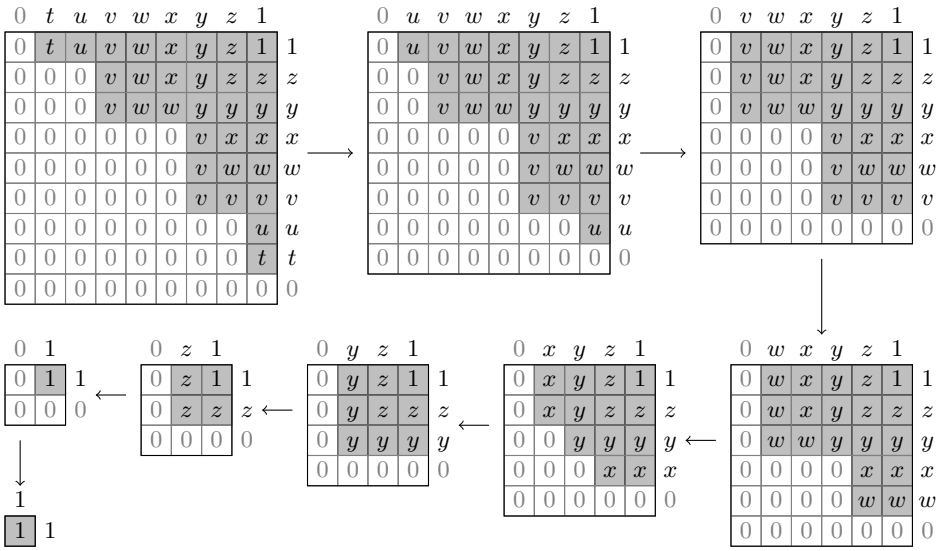

Fig. 2. Examples of f.n. tomonoids depicted by their Cayley tables. Seeing the cells of a table as ordered pairs from S^2, the level equivalence classes correspond with the maximal sets of the cells with the same symbol. The depicted f.n. tomonoids are actually created as one-element Rees quotients starting with the first f.n. tomonoid of size 9. As we may observe, the one-element Rees quotient arises by "cutting off" the column and the row indexed by the zero and by merging the zero and the atom classes into one. Finally, we reach the trivial monoid.

1. $\approx$ is a congruence [5] of S as a monoid and
2. each equivalence class is convex.

The operation induced by $\odot$ on the quotient $\langle S\rangle_\approx$ we denote again by $\odot$. For $a, b \in S$, we define $\langle a\rangle_\approx \leqslant \langle b\rangle_\approx$ if $a \approx b$ or $a < b$. We may observe that $(\langle S\rangle_\approx; \leqslant, \odot, \langle 1\rangle_\approx)$ is a tomonoid again and we call $\langle S\rangle_\approx$ the *tomonoid quotient* with respect to $\approx$. This procedure preserves the properties of finiteness, negativity, and commutativity.

We proceed with the notion of the Rees congruence which is commonly used for semigroups [6]. Let $q \in S$. For $a, b \in S$ we define $a \approx_q b$ if $a = b$ or $a, b \leqslant q$. Then $\approx_q$ is a tomonoid congruence and we call it the *Rees congruence* with respect to q. We denote the corresponding quotient by S/q and we call it the *Rees quotient* of S with respect to q. Furthermore, we call S a *Rees coextension* of S/q. If moreover $q = \alpha$, we call S/q the *one-element Rees quotient* of S and we call S the *one-element Rees coextension* (or, shortly, the *one-element coextension*) of S/q. See an illustration in Figure 2.

5 Theorem

The algorithm we are going to present is based on a theorem [9] which we briefly describe here. Let $(S; \leqslant, \odot, 1)$ be a f.n. tomonoid. We denote $S^\star = S \setminus \{0\}$. A *zero doubling extension* of S is a totally ordered set $\bar{S} = S^\star \dot\cup \{0, \alpha\}$ such that $0 < \alpha < a$ for every $a \in S^\star$. We call $a \in S$ an *idempotent* if $a \odot a = a$. Obviously, 0 and 1 are idempotents of every f.n. tomonoid.

Let $\sim_1$ and $\sim_2$ be two equivalence relations on S^2. We say that $\sim_2$ is a *coarsening* of $\sim_1$ if $\sim_1 \subseteq \sim_2$, that is, if each equivalence class of $\sim_2$ is a union of some equivalence classes of $\sim_1$.

Let $(S; \leqslant, \sim, (1,1))$ be a f.n. tomonoid partition. Let $\bar{S}$ be the zero doubling extension of S. Define

$$\mathcal{P} = \{(a,b) \in \bar{S}^2 \mid a, b \in S^\star \text{ and there is } c \in S^\star \text{such that } (a,b) \sim c\}, \tag{1}$$

$$\mathcal{Q} = \bar{S}^2 \setminus \mathcal{P}. \tag{2}$$

Let $(\varepsilon_l, \varepsilon_r)$ be a pair of non-zero idempotents of S and let $\dot\sim$ be the smallest equivalence relation on $\bar{S}^2$ such that the following conditions hold:

(E1) We have $(a,b) \dot\sim (c,d)$ for every $(a,b), (c,d) \in \mathcal{P}$ such that $(a,b) \sim (c,d)$.

(E2) We have $(d,c) \dot\sim (a,e)$ for every $(a,b), (b,c) \in \mathcal{P}$ and $(d,c), (a,e) \in \mathcal{Q}$ such that $(a,b) \sim d$ and $(b,c) \sim e$. (See an illustration in Figure 1-right.)

(E3'a) We have $(a,e) \dot\sim 0$ for every $a, b, c, e \in S^\star$ such that $(a,b) \in \mathcal{Q}$, $(b,c) \sim e$, and $c < \varepsilon_r$.
Furthermore, we have $(d,c) \dot\sim 0$ for any $a, b, c, d \in S^\star$ such that $(b,c) \in \mathcal{Q}$, $(a,b) \sim d$, and $a < \varepsilon_l$. (See an illustration in Figure 3-left.)

(E3'b) We have $(a,e) \dot\sim (a,b)$ for every $a, b, c, e \in S^\star$ such that $(a,b) \in \mathcal{Q}$, $(b,c) \sim e$, and $c \geq \varepsilon_r$.
Furthermore, we have $(d,c) \dot\sim (b,c)$ for every $a, b, c, d \in S^\star$ such that $(b,c) \in \mathcal{Q}$, $(a,b) \sim d$, and $a \geq \varepsilon_l$. (See illustrations in Figures 4-left and 4-right.)

(E3'c) We have $(a,b) \dot\sim 0$ for every $a, b, c > 0$ such that $(a,b), (b,c) \in \mathcal{Q}$, $a < \varepsilon_l$, and $c \geq \varepsilon_r$.
Furthermore, we have $(b,c) \dot\sim 0$ for every $a, b, c > 0$ such that $(a,b), (b,c) \in \mathcal{Q}$, $a \geq \varepsilon_l$, and $c < \varepsilon_r$. (See an illustration in Figure 3-right.)

(E4'a) We have $(1,0) \dot\sim (0,1) \dot\sim (a,\alpha) \dot\sim (\alpha,b)$ for every $a < \varepsilon_l$ and $b < \varepsilon_r$.
Furthermore, we have $(a,b) \dot\sim 0$ for every $(a,b), (c,d) \in \mathcal{Q}$ such that $(a,b) \trianglelefteq (c,d) \dot\sim 0$.

(E4'b) We have $(1,\alpha) \dot\sim (\alpha,1) \dot\sim (\varepsilon_l,\alpha) \dot\sim (\alpha,\varepsilon_r)$.
Furthermore, we have $(a,b) \dot\sim \alpha$ for every $(a,b), (c,d) \in \mathcal{Q}$ such that $(a,b) \trianglerighteq (c,d) \dot\sim \alpha$.

We call the structure $(\bar{S}^2; \trianglelefteq, \dot\sim, (1,1))$ the $(\varepsilon_l, \varepsilon_r)$-*ramification* of $(S^2; \trianglelefteq, \sim, (1,1))$.

Theorem 1. *[9] Let $(S; \trianglelefteq, \sim, (1,1))$ be a f.n. tomonoid partition and let $(\varepsilon_l, \varepsilon_r)$ be a pair of its non-zero idempotents. Let $(\bar{S}^2; \trianglelefteq, \dot\sim, (1,1))$ be the $(\varepsilon_l, \varepsilon_r)$-ramification of $(S^2; \trianglelefteq, \sim, (1,1))$.*

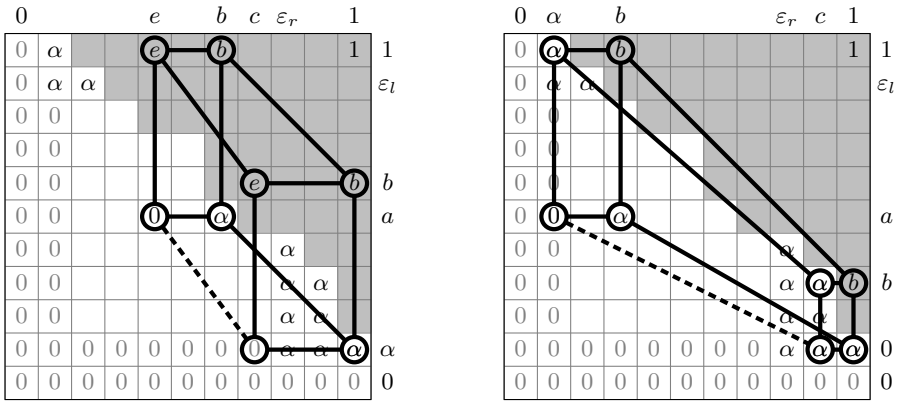

Fig. 3. Left: Illustration of (E3'a). Let $(a,b) \in \mathcal{Q}$, let $c < \varepsilon_r$, and let $(b,c) \sim e$. If $(a,b) \dot{\sim} 0$ then also $(a,e) \dot{\sim} 0$ according to the monotonicity. If $(a,b) \dot{\sim} \alpha$ then $(a,e) \dot{\sim} 0$ according to (P1). **Right: Illustration of (E3'c).** Let $(a,b),(b,c) \in \mathcal{Q}$, let $c \geq \varepsilon_r$, and let $a < \varepsilon_l$. Then $(a,b) \dot{\sim} 0$. Indeed, if we had $(a,b) \dot{\sim} \alpha$ then, according to (P1), we would also have $(a,\alpha) \dot{\sim} \alpha$ which is a contradiction.

If $(1,0) \dot{\sim} (1,\alpha)$ then there is no one-element coextension of S^2 with respect to $(\varepsilon_l, \varepsilon_r)$. Otherwise, let $\bar{\sim}$ be a coarsening of $\dot{\sim}$ such that the following holds: the $\bar{\sim}$-class of each $c \in S^\star$ coincides with the $\dot{\sim}$-class of c, the $\bar{\sim}$-class of 0 is downward closed, and each $\bar{\sim}$-class contains exactly one element of the form $(1,c)$ for some $c \in \bar{S}$. Then $(\bar{S}^2; \trianglelefteq, \bar{\sim}, (1,1))$ is a one-element coextension of S^2 with respect to $(\varepsilon_l, \varepsilon_r)$.

Moreover, all one-element coextensions of S^2 with respect to $(\varepsilon_l, \varepsilon_r)$, if there are any, arise in this way.

6 Representation of f.n. tomonoids

The crucial part, when implementing the algorithm, was to choose a suitable representation of the f.n. tomonoids (and the corresponding tomonoid partitions). F.n. tomonoids can be naturally represented by two-dimensional arrays representing their Cayley tables (see, e.g., Figure 2). However, this approach has shown as unsuitable for the implementation. Performing the algorithm, we mainly need to work with the level equivalence classes; we need, for example, to add pairs to this classes or we need to merge two classes into one.

Therefore we have decided to represent a f.n. tomonoid $(S; \leqslant, \odot, 1)$ as a collection of level equivalence classes of pairs from S^2. Such a collection forms a partition of S^2, i.e., every pair belongs to an (exactly one) equivalence class. Each level equivalence class is either assigned to a unique value of the f.n. tomonoid (which means that it must contain two pairs of the form $(a,1)$ and $(1,a)$ where $a \in S$) or it is "unassigned". An "unassigned" class can be a singleton.

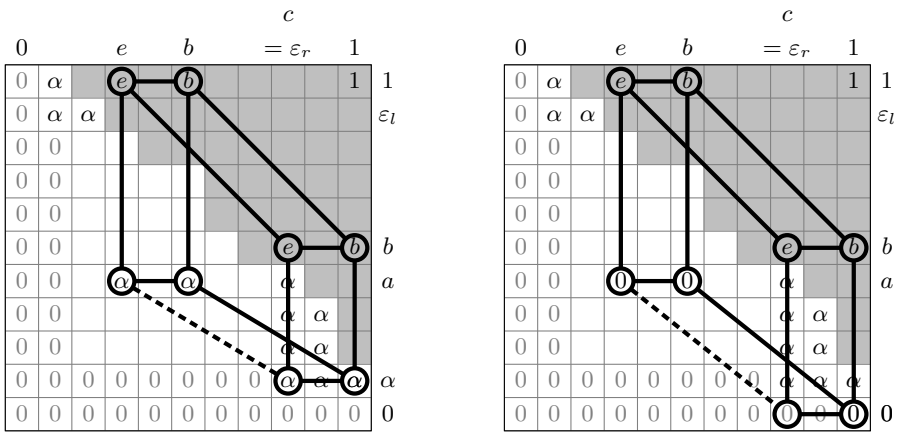

Fig. 4. Illustration of (E3'b). Left: Let $(a, b) \in \mathcal{Q}$, let $c \geqslant \varepsilon_r$, and let $(b, c) \sim e$. If $(a, b) \dot{\sim} \alpha$ then, according to (P1), $(a, e) \dot{\sim} \alpha$, as well. **Right:** Let $(a, b) \in \mathcal{Q}$, let $c \geqslant \varepsilon_r$, and let $(b, c) \sim e$. If $(a, b) \dot{\sim} 0$ then, according to (P1), $(a, e) \dot{\sim} 0$, as well.

7 Methods

Two methods, that recursively call each other, have been implemented:

- a method that adds a pair (a, b) to a z-level equivalence class, we denote it by $(a, b) \dot{\sim} z$,
- a method that relates a pair (a, b) with a pair (c, d), we denote it by $(a, b) \dot{\sim} (c, d)$.

When implementing these two methods, it is first crucial that the transitivity of $\dot{\sim}$ is preserved. That is, when we add a pair (a, b) to a certain level equivalence class, we consequently need to add to the same class also all the pairs that are already related to (a, b).

Second, it is important that the monotonicity of the constructed tomonoid is not violated. This task is easier by the fact that (except for Part (E1), see below) we work only with pairs that are assigned to 0, α, or unassigned. Thus, when performing $(a, b) \dot{\sim} z$, z is either 0 or α. If $z = 0$ then we need to be sure that also all the pairs lower that (a, b) are assigned to 0. If $z = \alpha$ we proceed analogously for the pairs greater that (a, b). The details are described in the next two subsections.

Method implementing $(a, b) \dot{\sim} z$

Recall that z is either 0 or α. We delete the whole level equivalence class containing (a, b) and we add all the deleted pairs to the z-level equivalence class. If $z = 0$ then for every pair (x, y) in the deleted class:

– for every pair $(u,v) \in \mathcal{Q}$ such that $(u,v) \trianglelefteq (x,y)$:
  • perform $(u,v) \dot{\sim} 0$.

If $z = \alpha$ then for every pair (x,y) in the deleted class:

– for every pair $(u,v) \in \mathcal{Q}$ such that $(u,v) \trianglerighteq (x,y)$:
  • perform $(u,v) \dot{\sim} \alpha$.

If (a,b) is already contained in a y-level equivalence class and $y \neq z$ an error is emitted signalizing that the constructed coextension is not possible.

Method implementing $(a,b) \dot{\sim} (c,d)$

If both the pairs (a,b) and (c,d) belong to unassigned level equivalence classes, we simply delete one of the classes and add all its pairs to the second one.

If one of the pairs, say (a,b), belongs to a z-level equivalence class (z is either 0 or α), we perform $(c,d) \dot{\sim} z$.

If (a,b) belongs to a z-level equivalence class and (c,d) belongs to a y-level equivalence class then either $y = z$ which means that both (a,b) and (c,d) belong to the same level equivalence class and thus we do not perform anything, or $y \neq z$ which means that it is not possible to construct such a coextension. In the latter case an error is emitted stopping the process.

8 Algorithm

Input:

– $(S^2; \trianglelefteq, \sim, (1,1))$ ... tomonoid partition of a f.n. tomonoid $(S; \leqslant, \odot, 1)$
– $(\varepsilon_l, \varepsilon_r)$ ... pair of its non-zero idempotents

Output:

– $(\bar{S}^2; \trianglelefteq, \bar{\sim}, (1,1))$ ... a one-element coextension of $(S^2; \trianglelefteq, \sim, (1,1))$ with respect to $(\varepsilon_l, \varepsilon_r)$

Algorithm:

Initialization:

1. Let $\bar{S}$ be the zero doubling extension of S.
2. Let 0, α, and κ be the zero, the atom, and the coatom of $\bar{S}$, respectively. Let $\mathcal{P}$ and $\mathcal{Q}$ be given by (1) and (2), respectively.
3. Let $\dot{\sim}$ be an equivalence relation on $\bar{S}^2$. (The following steps are going to define this relation.)

Part (E1):

4. For every $(a,b),(c,d) \in \mathcal{P}$:

- define $(a,b) \dot{\sim} (c,d)$
  if $(a,b) \sim (c,d) \sim e$ for some $e \in \bar{S} \setminus \{0, \alpha\}$.

Part (E2):

5. For every $(a,b),(b,c) \in \mathcal{P}$:
   - let $d \in \bar{S}$ be such that $(a,b) \sim d$,
   - let $e \in \bar{S}$ be such that $(b,c) \sim e$,
   - perform $(a,e) \dot{\sim} (d,c)$.

Part (E4'):

6. Perform $(1,0) \dot{\sim} (0,1) \dot{\sim} 0$.
7. Perform $(a,\alpha) \dot{\sim} (\alpha,b) \dot{\sim} 0$ for $a < \varepsilon_l$ and $b < \varepsilon_r$.
8. Perform $(\varepsilon_l,\alpha) \dot{\sim} (\alpha,\varepsilon_r) \dot{\sim} \alpha$.

Part (E3'a):

9. For every $a \in \bar{S}$ such that $\alpha < a < \varepsilon_l$:
   - let $b \in \bar{S}$ be the highest element such that $(a,b) \in \mathcal{Q}$,
   - let $c \in \bar{S}$ be the highest element such that $c < \varepsilon_r$,
   - let $e \in \bar{S}$ be such that $(b,c) \sim e$,
   - if $e > \alpha$ then perform $(a,e) \dot{\sim} 0$.
10. For every $c \in \bar{S}$ such that $\alpha < c < \varepsilon_r$:
    - let $b \in \bar{S}$ be the highest element such that $(b,c) \in \mathcal{Q}$,
    - let $a \in \bar{S}$ be the highest element such that $a < \varepsilon_l$,
    - let $d \in \bar{S}$ be such that $(a,b) \sim d$,
    - if $d > \alpha$ then perform $(d,c) \dot{\sim} 0$.

Part (E3'c):

11. For every $a \in \bar{S}$ such that $\varepsilon_l \leqslant a < 1$:
    - let $b \in \bar{S}$ be the highest element such that $(a,b) \in \mathcal{Q}$,
    - let $c \in \bar{S}$ be the highest element such that $(b,c) \in \mathcal{Q}$ and $c < \varepsilon_r$,
    - perform $(b,c) \dot{\sim} 0$.
12. For every $c \in \bar{S}$ such that $\varepsilon_r \leqslant c < 1$:
    - let $b \in \bar{S}$ be the highest element such that $(b,c) \in \mathcal{Q}$,
    - let $a \in \bar{S}$ be the highest element such that $(a,b) \in \mathcal{Q}$ and $a < \varepsilon_l$,
    - perform $(a,b) \dot{\sim} 0$.

Part (E3'b):

13. For every $b \in \bar{S}$ such that $\alpha < b < 1$:
    - let $e \in \bar{S}$ be such that $(b,\varepsilon_r) \sim e$,
    - if $e < b$ then:
      - for every $a \in \bar{S}$ s.t. $\alpha < a < \varepsilon_l$ and $(a,b) \in \mathcal{Q}$:
        - perform $(a,e) \dot{\sim} (a,b)$.
14. For every $b \in \bar{S}$ such that $\alpha < b < 1$:
    - let $d \in \bar{S}$ be such that $(\varepsilon_l,b) \sim d$,

- if $d < b$ then:
  - for every $c \in \bar{S}$ s.t. $\alpha < c < \varepsilon_r$ and $(b, c) \in \mathcal{Q}$:
    - perform $(d, c) \dot{\sim} (b, c)$.

Coarsening:

15. Let $\tilde{\sim} := \dot{\sim}$.
16. For every pair $(a, b) \in \bar{S}^2$, that belongs to an unassigned level equivalence class, perform arbitrarily either $(a, b) \tilde{\sim} 0$ or $(a, b) \tilde{\sim} \alpha$.

Remark 1. Let $\varphi \in S$ be the lowest non-zero idempotent of S. In Step 5 we may omit those pairs $(a, b), (b, c) \in \mathcal{P}$ where $(a, b), (b, c) \trianglerighteq (\varphi, \varphi)$ since, in such a case, $(a, e), (d, c) \in \mathcal{P}$.

Remark 2. In order to obtain all the one-element coextensions of S we simply repeat the procedure for every possible pair of its non-zero idempotents including $(1, 1)$. Furthermore, we create an additional coextension in the following way:

- Perform Steps 1 and 2.
- Perform $(1, 0) \tilde{\sim} (0, 1) \tilde{\sim} 0$.
- Perform $(\alpha, \alpha) \tilde{\sim} \alpha$.

9 Example

Let us perform the algorithm taking the first f.n. tomonoid of size 9 in Figure 2. As we can see, it has three non-zero idempotents: y, z, and 1. We are going to construct all the one-element coextensions with respect to (z, y).

- Initialization, Part (E1), and Part (E4'):
  - We obtain the values depicted in Figure 5-a.
- Part (E3'a) (see Figure 5-b):
  - Step 9:
    - For $(b, c) = (t, z)$ we perform $(y, t) \dot{\sim} 0$.
    - For $(b, c) = (u, z)$ we perform $(y, u) \dot{\sim} 0$.
    - For $(b, c) = (v, x)$ we perform $(v, v) \dot{\sim} 0$.
    - For $(b, c) = (w, x)$ we perform $(v, w) \dot{\sim} 0$.
    - For $(b, c) = (x, x)$ we perform $(v, x) \dot{\sim} 0$.
    - For $(b, c) \in \{(y, u), (u, z)\}$ we do not perform anything.
  - Step 10:
    - for $(a, b) = (z, t)$ we perform $(x, t) \dot{\sim} 0$.
    - for $(a, b) = (z, u)$ we perform $(x, u) \dot{\sim} 0$.
    - for $(a, b) \in \{(x, v), (x, w), (x, x), (u, y), (x, z)\}$ we do not perform anything.
- Part (E3'c) (see Figure 5-c):
  - Step 11:
    - For $a = y$ we obtain $b = u$ and $c = y$. Thus we perform $(u, y) \dot{\sim} 0$.
    - For $a = z$ we obtain $b = u$ and $c = y$. Thus we perform $(u, y) \dot{\sim} 0$.

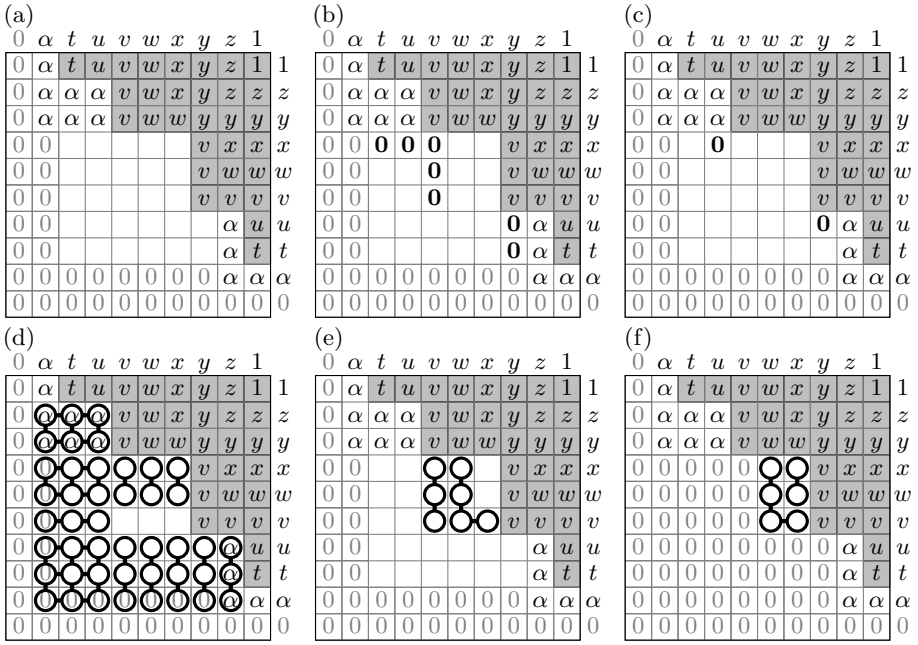

Fig. 5. Illustration of the algorithm.

- Step 12:
  - For $c = z$ we obtain $b = u$ and $a = x$. Thus we perform $(x, u) \dot{\sim} 0$.

– Part (E3'b) (see Figure 5-d):
- Step 13:
  - For $b = t$ we obtain $e = \alpha$ and we perform $(a, \alpha) \dot{\sim} (a, t)$ for every a from α to x.
  - For $b = u$ we obtain $e = \alpha$ and we perform $(a, \alpha) \dot{\sim} (a, u)$ for every a from α to x.
- Step 14:
  - For $b = t$ we obtain $d = \alpha$ and we perform $(\alpha, c) \dot{\sim} (t, c)$ for every c from α to y.
  - For $b = u$ we obtain $d = \alpha$ and we perform $(\alpha, c) \dot{\sim} (u, c)$ for every c from α to y.
  - For $b = x$ we obtain $d = w$ and we perform $(w, c) \dot{\sim} (x, c)$ for every c from α to x.
  - For $b = z$ we obtain $d = y$ and we perform $(y, c) \dot{\sim} (z, c)$ for every c from α to u.

– Part (E2) (see Figure 5-e):
- For $a = x$, $b = y$, and $c = x$ perform $(x, w) \dot{\sim} (v, x)$.
- For $a = w$, $b = y$, and $c = x$ perform $(w, w) \dot{\sim} (v, x)$.
- For $a = v$, $b = y$, and $c = x$ perform $(v, w) \dot{\sim} (v, x)$.

  - For $a = x$, $b = y$, and $c = w$ perform $(x, w) \dot{\sim} (v, w)$.
  - For $a = w$, $b = y$, and $c = w$ perform $(w, w) \dot{\sim} (v, w)$.
  - For $a = x$, $b = y$, and $c = v$ perform $(x, v) \dot{\sim} (v, v)$.
  - For $a = w$, $b = y$, and $c = v$ perform $(w, v) \dot{\sim} (v, v)$.
- Finally, we obtain the situation depicted in Figure 5-f. As we can see, there are three distinct one-element coextensions of the tomonoid.

10 Conclusion

All the steps of the algorithm run in a polynomial time (with respect to the size of S) except for Step 16 where we, actually, obtain all the possible one-element coextensions with respect to the given pair of idempotents $(\varepsilon_l, \varepsilon_r)$. This step runs in exponential time since also the number of the coextensions is exponential, in the worst case.

If we wish to obtain all the commutative one-element coextensions of a commutative f.n. tomonoid S, we simply perform $(a, b) \dot{\sim} (b, a)$ for every $(a, b) \in \mathcal{Q}$ right after Initialization.

The algorithm has been implemented and tested in the programming language Python [8].

References

1. J. Aczél, Quasigroups, nets and nomograms, *Advances in Mathematics* **1** (1965), 383–450.
2. W. Blaschke, G. Bol, "Geometrie der Gewebe, topologische Fragen der Differentialgeometrie" (in German), Springer, Berlin 1939.
3. F. Esteva, L. Godo, Monoidal t-norm based logic: towards a logic for left-continuous t-norms, *Fuzzy Sets and Systems* **124** (2001), 271–288.
4. K. Evans, M. Konikoff, J. J. Madden, R. Mathis, and G. Whipple, Totally ordered commutative monoids, *Semigroup Forum* **62** (2001), 249–278.
5. P. A. Grillet, "Semigroups. An introduction to the structure theory", Marcel Dekker, New York 1995.
6. J. M. Howie, "An introduction to semigroup theory", Academic Press, London 1976.
7. M. Petrík and Th. Vetterlein, Algorithm to generate the Archimedean, finite, negative tomonoids. In *Joint 7th International Conference on Soft Computing and Intelligent Systems and 15th International Symposium on Advanced Intelligent Systems*, pages 42–47, Kitakyushu, Japan, December 2014.
8. M. Petrík and Th. Vetterlein, Program to find elementary extensions of finite, negative, totally ordered monoids, *source code written in Python*, 2015, downloadable at `http://cmp.felk.cvut.cz/~petrikm/extensions.php`.
9. M. Petrík and Th. Vetterlein, Rees coextensions of finite, negative tomonoids, *Journal of Logic and Computation*, (2015). DOI: 10.1093/logcom/exv047.