Matrix Representation of the Verification of Weak and Easy Soundness of Robustness Diagram with Loops and Time Contol

Nathaniel Enrique C. Eulin

November 24, 2024

Contents

Chapter 1

Introduction

1.1 Background of the Study

A workflow consists of three dimensions, namely the process, resource, and case dimension [1]. The process dimension is a specification of a process, the process being a partial ordering of a set of tasks such as control schemes like sequential, conditional, or iteration. The resource dimension pertains to the resource specification, where a resource is an object in the system that performs calculations, or processes, a task. Finally, the case dimension is the specification of the case, where case refers to the abstraction of a set of entities executed according to how the process is defined by the proper resource. [2] [5].

Workflow models represent complex and dynamic real-world systems in various fields such as human resource management, biology (integrated disease surveillance [Lopez et al. 2020]), and manufacturing engineering (chiller systems [8]). This includes models such as Petri nets (PN) and Robustness Diagrams with Loop and Time Controls (RDLT) [5]. These models allow workflow components to be analyzed and model properties to be verified.

The study of the workflow model Robustness Diagrams with Loop and Time Controls has been important in recent years due to its capability to represent a workflow in all of its dimensions (process, resource, and case). Although there are other workflow models

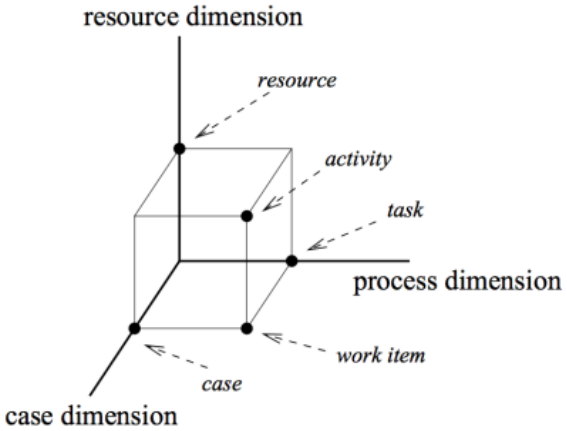

Figure 1.1: The three dimensions of workflows. (Image source:[5])

that can represent all dimensions e.g. business process modeling and notation (BPMN), activity diagrams, etc., there are gaps and challenges in verifying the properties of these models. Issues like concept excess, lack of support for explicit representation of data and rules, functional decomposition, etc. The dynamicity of RDLT allows it to be mapped to different mathematical models which enables different ways for the verification of model properties, such as soundness. [5]

Classical soundness, in particular, refers to the property of a workflow to be absent of livelocks, deadlocks, and other anomalies in the flow [8]. In general, to be classically sound, a model should have proper termination and liveness. There are other notions of soundness derived from being classically sound, namely, relaxed, weak, easy, and lazy soundness. Relaxed soundness [6] is already defined as well as the easy, weak, and lazy notions of soundness [8], all of which differ in the degree of relaxations of proper termination and liveness [6].

Hence, with all the knowledge about workflows and RDLT's notions of soundness, further research would expand in terms of automation to verify model properties. Therefore, various research has been done about the representation of RDLT into mathematical models aiming to automize verification. In Karen and Roben's work [4], they proposed a matrix representation of RDLT to verify soundness. Asoy [6] used a matrix representation of RDLT to verify and automate classical soundness. This paper aims to leverage the structural behaviors of easy and weak soundness as formally defined by Ramirez (2024) and use matrix representations to verify these notions of soundness in an RDLT.

1.2 Basic Definitions and Notations

Robustness Diagram with Loop and Time Controls (RDLT)

In this section, RDLT and its notions of soundness is defined. RLDT is a workflow model that allows all three dimension of a workflow to be represented.

Definition 1.2.1. RDLT [5]
An RDLT is a graph representation R of a system that is defined as $R = (V, E, T, M)$ where:

- V is a finite set of vertices, where each vertex has a type $V_{type} : V \to \{$'b', 'e', 'c'$\}$ where 'b', 'e', and 'c' means the vertex is either a "boundary object", an "entity object", or a "controller", respectively.
- A finite set of arcs $E \subseteq (V \times V) \backslash E'$ where $E' = \{(x, y) | x, y \in V, V_{type}(x) \in \{$'b', 'e'$\}, V_{type}(y) \in \{$'b', 'e'$\}\}$ with the following attributes with user-defined values,
  - $C : E \to \Sigma \cup \{\epsilon\}$ where Σ is a finite non-empty set of symbols and ϵ is the empty string. Note that for real-world systems, a task $v \in V$, i.e. $V_{type}(v) =$'c', is executed by a component $u \in V, V_{type}(u) \in \{$'b','e'$\}$. This component-task association is represented by the arc $(u, v) \in E$ where $C((u, v)) = \epsilon$. Furthermore, $C((x, y)) \in \Sigma$ represents a constraint to be satisfied to reach y from x. This constraint can represent either an input requirement or a parameter $C((x, y))$ which needs to be satisfied to proceed from using the

component/task x to y. $C((x, y)) = \epsilon$ represents a constraint-free process flow to reach y from x or a self-loop when $x = y$.

- $L : E \to \mathbb{Z}$ is the maximum number of traversals allowed on the arc.

- Let T be a mapping such that $T((x, y)) = (t1, ..., tn)$ for every $(x, y) \in E$ where $n = L((x, y))$ and $t_i \in \mathbb{N}$ is the time a check or traversal is done on (x, y) by some algorithm's walk on R.

- $M : V \to \{0, 1\}$ indicates whether $u \in V$ and every $v \in V$ where $(u, v) \in E$ and $C((u, v)) = \epsilon$ induce a sub-graph G_u of R known as a **reset-bound subsystem** (RBS). The RBS G_u is induced with the said vertices when $M(u) = 1$. In this case, u is referred to as the **center** of the RBS G_u. G_u's vertex set V_{G_u} contains u and every such v, and its arc set E_{G_u} has $(x, y) \in E$ if $x, y \in V_{G_u}$.

  Finally, $(a, b) \in E$ is called an **in-bridge** of b if $a \notin V_{G_u}, b \in V_{G_u}$. Meanwhile, $(b, a) \in E$ is called an **out-bridge** of b if $b \in V_{G_u}$ and $a \notin V_{G_u}$. Arcs $(a, b), (c, d) \in E$ are **type-alike** if $\exists y \in V$ where $(a, b), (c, d) \in Bridges(y)$ with $Bridges(y) = \{(r, s) \in E | (r, s)$ is either an in-bridge or out-bridge of $y\}$ or if $\forall y \in V, (a, b), (c, d) \notin Bridges(y)$.

Figure 1.2 shows an RLDT with a reset-bound system with center x_2 and its owned controllers x_3 and x_4. The in-bridges of the RBS with respect to x_2 are (x_1, x_2) and (x_6, x_2). Both arcs are type-alike with respect to x_2 as defined in 1.2.1. However, (x_3, x_2), being inside the RBS, is not type-alike to any of the aforementioned arcs. (x_4, x_5) and (x_4, x_6) are type-alike with respect to x_4. For each arcs of this RDLT, $C-$ and $L-$ attributes are defined. The x_1 serves as the source, while the x_7 serves as the sink.

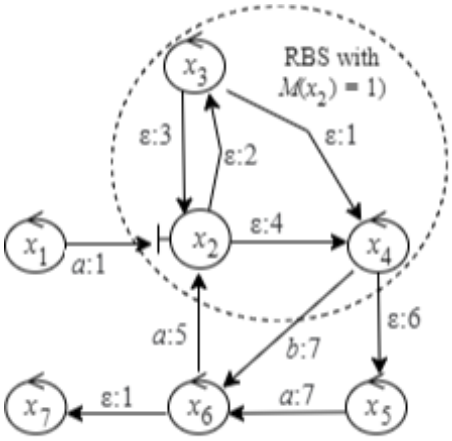

Figure 1.2: A Reset-Bound Subsystem-Containing Robustness Diagram with Loop and Time Controls with center at x_2, where $M(x_2) = 1$. (Image source: [7])

Activity Extraction Algorithm

The proposed Activity Extraction algorithm in [5] takes an input RDLT R with a defined start index s and goal vertex f. As shown in Algorithm 1 it returns an activity

profile defined in 1.2.3, a set of exected vertices from the extracted activities. To understand activity extraction fully, along with the definition of activity profile, reachability configuration, defined in 1.2.2, is also first introduced.

Definition 1.2.2. ***Reachability Configuration***
A reachability configuration $S(t)$ in $R = (V, E, \Sigma, C, L, M)$ contains the arcs traversed by $\mathcal{A}$ at time step $t \in \mathbb{N}$.

Definition 1.2.3. ***Activity Profile***
A set $S = \{S(1), S(2), \ldots, S(d)\}$ of reachability configurations, $d \in \mathbb{N}$, is an activity profile in $R = (V, E, \Sigma, C, L, M)$ where $\exists (u, v) \in S(1)$ and $(x, y) \in S(d)$ such that $\nexists w, z \in V$ where $(w, u), (y, z) \in E$.

Another definition, that of *Unconstrained Arc* in 1.2.4 defines what an unconstrained arc is which allows for the traversal of arcs in activity extraction.

Algorithm 1 Activity Extraction Algorithm $\mathcal{A}$ [5, 9]

Input: R, $s \in V$, $f \in V$
Output: vertices S of R, $\emptyset$ otherwise

Initialize S
for every (x, y) **do**
    Initialize $T((x, y))$ such that $T((x, y)) = (t_1, ..., t_n)$ where $n = L((x, y))$ and $t_i \epsilon \mathbb{N}$ is the time a check or traversal is done on (x, y) by A.
end for
Let $x = s$.
while $x \neq f$ **do**
    Select $(x, y) \epsilon E$ where $L((x, y))$ has not been reached.
    if $\exists (u, x) \epsilon E = max(T((u, x)))$ **then**
        Assign $maxV + 1$ to the leftmost zero of $T((x, y))$.
    else
        $maxV = 0$.
    end if
    Determine whether (x, y) is an unconstrained arc or not.
    if (x, y) is unconstrained **then**
        Traverse (x, y) .
        Assign $MAX + 1$ to $T((x, y))$ where MAX is the maximum value from all $T((v', y)) \; \forall \; v' \epsilon V$ where $(v', y), (v, y)$, and (x, y) are type-alike.
        for every $(v, y) \epsilon E$ that is type-alike with (x, y) **do**
            Assign $MAX + 1$ to every $T((x, y))$.
            if $C((v, y)) \epsilon \Sigma$ **then**
                The last value in $T((x, y))$ where the last check was done on (v, y) is updated.
            else if v is either type 'b' or 'e' and y is type 'c' **then**
                The first value in $T((x, y))$ is updated.
            end if
        end for
    else if (x, y) is not unconstrained and not other $(x, y') \epsilon E$ where $y' \epsilon V$ can be selected **then**
        Backtrack to $a \epsilon V$ where $(a, x) \epsilon E$ and a was previously visited by the algorithm to reach x.
    end if
end while
if activity extraction fails **then**
    return $\emptyset$;
else
    return S
end if

In step 4.3 of algorithm $\mathcal{A}$, an evaluation is performed to determine if $(x, y)\epsilon E$ is an unconstrained arc. The definition of an unconstrained arc is defined as follows:

Definition 1.2.4. ***Unconstrained Arc***
An arc $(x, y) \in E$ is **unconstrained** if $\forall(v, y) \in E$, where (x, y) and (v, y) are type-alike, any of the following traversal conditions hold,

1. $C((v, y)) \in \{\epsilon, C((x, y))\}$,
2. $|\{t_i \in T((x, y))|t_i \geq 1\}| \leq |\{t_j \in T((v, y))|t_j \geq 1\}| \leq L((v, y))$,
3. $C((v, y)) \in \Sigma, C((x, y)) = \epsilon \wedge T(v, y) \neq [0]$.

Note that (x, y) will remain unconstrained (if it is such) regardless of any other (v, y) where (x, y) and (v, y) are not type-alike.

Vertex Simplification

Definition 1.2.5. Vertex Simplified R [5]
A vertex-simplified RDLT $G = (V', E', C')$ of $R = (V, E, T, M)$ (with arc attributes C and L) is a multigraph whose vertices $v \in$ V have $V_{type}(v) =' c'$ where G is derived from R such that the following holds:

1. $x \in V'$ if any of the following holds:
   - $x \in V$ and $x \notin V_{G_u}$ of an RBS G_u in R, or
   - there exists an in-bridge $(q, x) \in E$ of $x \in V \cap V_{G_u}, q \in V$ of R, or
   - there exists an out-bridge $(x, q) \in E$ of $x \in V \cap V_{G_u}, q \in V$ of R, or
2. $(x, y) \in E'$ with $C'((x, y)) = C((x, y))$ for $x, y \in V'$ if $(x, y) \in E$
3. $C((x, y)) = \varepsilon$ if $x, y \in V' \cap V_{G_u}$ and x is an ancestor of y in R and $(x, y) \notin E_{G_u}$

We refer to this simplification of R as **level-1 vertex simplification** of R with respect to every RBS G_u in R. A **level-2 vertex simplification** of R with respect to its RBS G_u is the level-1 vertex-simplification of G_u where G_u is treated as an RDLT where the value of the vertex attribute M of u is redefined to 0, i.e. $M(u) = 0$. With this, the verification of model properties (i.e. maximally composed and sound RDLTs) are separately done for the level-1 and level-2 vertex simplifications of R. However, this separation does not affect the validity of proving for any of these properties on the entire RDLT itself. Furthermore,

In simpler terms, the vertices in R present in a vertex-simplified RDLT G are those that do not belong in any RBS or those inside an RBS but has an *in-bridge* and/or an *out-bridge*. The vertex-simplifications R_1 and R_2 of the RDLT in 1.2 is shown in 1.3, respectively. R_1 captures a subgraph of R found outside of its RBS, including the vertices of the RBS that have at least 1 in-brigde or out-bridge. R_2 vertex-simplification, however, captures the RBS itself. The vertices that have in- or out-bridges in R are marked either as sources and/or sinks in R_2.

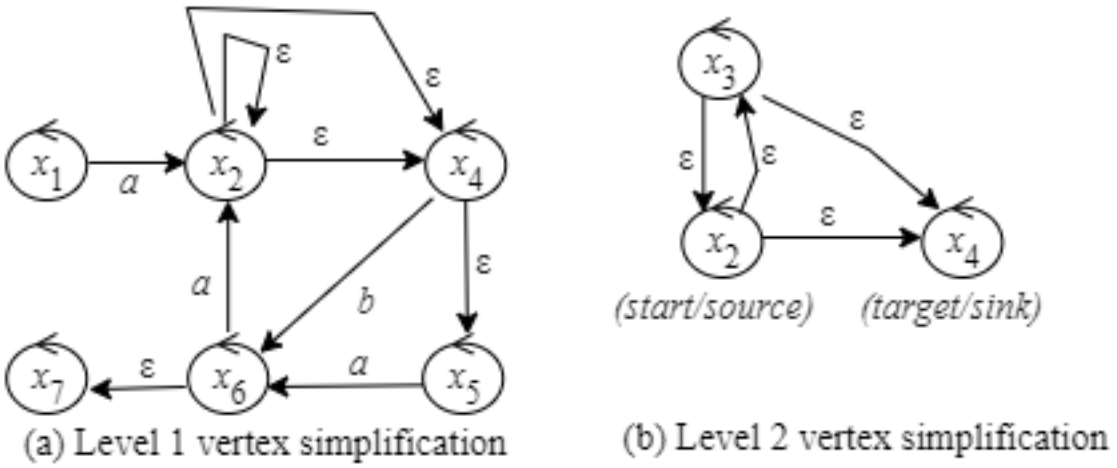

Figure 1.3: The levels 1 and 2 vertex simplification R_1 and R_2 of the RLDT in 1.2 (Image Source: [7])

L-Safeness of RDLTs

In the analysis of processes in RDLT, particularly on its verification of soundness property, it is crucial that possible L-attrbute-based deadlocks are avoided. Reachability of the vertices relies heavily on the configuration of the arcs' $L-$values. The following definitions discusses the $L-$safeness with RDLTs from [6].

Definition 1.2.6. Critical and Escape Arcs [6] A cycle $c = [x_1 x_2 \dots x_n]$ is a sequence of vertices $x_i \in V$, where $(x_i, x_{i+1}) \in E$, $i = 1, 2, \dots, n-1$, and no two vertices in c are the same except for $x_1 = x_n$. The elements of c are denoted as $Lit(c)$. We denote the set of arcs of c as $ArcsOfCycle(c) = \{(x, y) \in E | \exists x_i, x_{i+1} \in Lit(c)$ where $x = x_i$ and $y = x_{i+1}\}$.

$(x, y) \in ArcsOfCycle(c)$ is called a critical arc (CA) in c if it has the minimum L-value among the arcs in c, i.e. $L(x, y) = min_{(u,v)\in ArcsOfCycle(c)}\{L(u, v)\}$. If $\exists (x, v) \in E$ such that $v \in V \backslash Lit(c)$, then we refer to (x, v) as an escape arc (EA) of (x, y) in c. Self-loops, i.e. (x, x), are cycles that are themselves entirely composed of one critical arc that does not affect the (re)use of other arcs in an RDLT.

Definition 1.2.7. Loop-Safe Arcs [6] Let R be a connected RDLT, i.e. for every vertex $v \in V$, there is a path from a source vertex $s \in V$ to v in R. A non-critical arc (NCA) $(x, y) \in E$ of R is loop-safe if $L(x, y) > RU(x, y)$, where

$$RU(x, y) = \Sigma_{k=1^{|Cycles(x,y)|}}(I * L(u, v)), \text{ for some } (u, v) \in minL\_CA(c_k), \text{ with}$$
$$I = \begin{cases} 1, & \text{if } k = 1 \text{ or } \cup_{j=1^{k-1}} minL\_CA(c_j) \cap minL\_CA(c_k) \\ 0, & \text{otherwise,} \end{cases}$$

and $minL\_CA(c) \subset E$ is the set of arcs whose L-value is the minimum among the critical arcs found in c, accounting the other cycles in c' in R that intersect with c.

Definition 1.2.8. Safe Critical Arcs [6] A CA (x, y) *in* E is safe if there is an escape, non-critical arc (x, z) *in* E, where (x, z) is loop-safe.

JOINs in RDLTs

In RDLTs, it is not impossible for multiple processes to converge at a single vertex (JOIN). The type of join specifies how the incoming processes interact before proceeding in the workflow. The following definitions are the different types of joins in RDLTs:

1. AND-JOIN at $y \in V$: For every type-alike arcs $(v, y), (u, y) \in E$ where $(v, y) \neq (u, y)$, their C-values $C(v, y) \neq C(u, y)$ and $C(v, y), C(u, y) \in \Sigma$.

2. MIX-JOIN at $y \in V$: There is at least one pair of type-alike arcs $(v, y), (u, y) \in E$ where $(v, y) \neq (u, y)$, $C(v, y) \neq C(u, y)$ and $C(v, y) = \epsilon$, and $C(u, y) \in \Sigma$.

3. OR-JOIN: For every type-alike arcs $(v, y), (u, y) \in E$ where $C(v, y) = C(u, y)$.

The concept of JOIN-safe L-values ensures processes are in proper coordination with each other. This, along with 1.2.7 and 1.2.8, provides the structural requirement for a classically sound RDLT, e.g, proper termination and liveness. These definitions serves as the basis for determining looser notions of soundness.

Definition 1.2.9. JOIN-safe L-values [6] For every pair of arcs (u, y), $(v, y) \in E$, where $C(u, y) \neq C(v, y)$, we say that (u, y) and (v, y) have JOIN-safe L-values if:

1. **One split origin.** There is exactly one common ancestor $x \in V$ for u and v such that there is exactly one path $P_u = a_1 a_2 \dots a_n$, where $a_1 = x$, $a_{n-1} = u$, $a_n = y$, $a_i \in V$, $1 < I < n - 2$, and another path $P_v = b_1 b_2 \dots b_m$, where $b_1 = x$, $b_{m-1} = v$, $b_m = y$, $b_j \in V$, $1 < j < m - 2$. Furthermore, P_u and P_v do not intersect with each other except at x and y. That is, no a_i and b_j along these paths are equal, for $1 < i < n$ $1 < j < m$; and

2. **No unrelated process.** For every arc $(a, b) \in E$, if $a = x$, where x is the one common ancestor of u and v, then there is no path from a to another vertex $r \in V$ such that for some $(r, s) \in E$, $s \neq y$.

3. **No branching out from every related process.** $\forall q_k \in Lit(P_q)$, where $q \in \{u, v\}$, $1 \leq k < |Lit(P_q)|$, $\nexists(q_{k,s}) \in E$ where $s \in V \backslash Lit(P_q)$.

4. **No process interruptions.** $\forall q_k \in Lit(P_q)$, where $q \in \{u, v\}$, $1 < k \leq |Lit(P_q)|$, $\nexists(s, q_k) \in E$ where $s \in V \backslash Lit(P_q)$.

5. **Duplicate conditions.**

   - If $C(u, y)$, $C(v, y) \in \Sigma$, i.e. an AND-JOIN at y, then there is no process $P = [x_1 x_2 \dots x_p]$ in R, where $x_1 = x$, $x_p = y$, such that $C(x_{p-1}, x_p) = C(u, y)$ (or $C(v, y)$).
   - Without any loss of generality, if $C(u, y) = \varepsilon$ and $C(v, y) \in \Sigma$, then any process $P = x_1 x_2 \dots x_p$ in R, where $x_1 = x$, $x_p = y$, can have $C(x_{p-1}, x_p) \in \{\varepsilon, C(v, y)\}$. That is, a MIX-JOIN can have duplicate conditions for the arcs connecting to y, but no two of such arcs have different conditions.

6. **Equal L-values of arcs at the AND-JOIN.**

7. **Loop-safe components of every related process.** For an AND- or MIX-JOIN merging at y, each of its processes $P = x_1x_2 \dots x_k$, $k \in \mathbb{N}$, where $x_1 = x$ and $x_k = y$, the arc $(x_1, x_{i+1}) \in E$. $i = 1, 2, \dots, k-1$, is loop-safe. Lastly, for an OR-JOIN merging at y, each process $P = x_1x_2 \dots x_k$, $k \in \mathbb{N}$, of this JOIN has its arc (x_i, x_{i+1}) to have a JOIN-safe L-value, $i = 1, 2, \dots, k-1$, if (x_i, x_{i+1}) is either a loop-safe arc or a safe CA in R.

Reusability and Resets in RDLTs

The concept of reusability refers to how components of an RDLT can be used in different situations, such as when an RDLT has an RBS [7]. The information in this section is sourced from [7] unless explicitly mentioned otherwise.

Definition 1.2.10. Pseudocritical Arcs, Pseudo-escape Arcs
A pseudocritical arc $(PCA)(x, y) \in E$, where (x, y) is a component of a cycle c in R where $L(x, y)$ is the minimum among all the L-values of the arcs in c which are not arcs of an RBS in R.

Meanwhile, a pseudo-escape arc $(PEA)(x, z) \in E$ is a non-critical arc in R where (x, y) and (x, z) are type-alike.

Definition 1.2.11. Expanded Reusability in RDLTs with Resets
Let $R = (V, E, T, M)$ be a connected $RDLT$. If $\exists v \in V$ where $M(v) = 1$, then let $B = (V', E', T', M')$ be the RBS with its center v.

1. Let $IB_r(X)$ be the set of in-bridges of $x \in V$.
2. Let $Cycles_{part}(R)$ be a set of cycles in R with the following property: $\forall p = [x_1x_2...x_n] \in Cycles_{part}(R)$, there exist cycle components meeting these conditions:
   (a) $(x_i, x_{i+1}) \in E$ is inside an RBS B of R (i.e, $x_i, x_{i+1} \in V'$).
   (b) $(x_j, x_{j+1}) \in E$ is not inside B (i.e, $x_j or x_{j+1} is not in V'$).

Whenever we have cycles $d = [d_1d_2...d_{m1}], e = [e_1e_2...e_{m1}] \in Cycles_{part}(R)$ that overlap in their non-RBS components, and this overlap contains both of their PCAs $(d_k, d_{k+1}), (e_{k'}, e_{k'+1})$, where $L(d_k, d_{k+1}) \geq L(e_{k'}, e_{k'+1})$, then consider only one of these PCAs. Choose the one with the minimum L-value as it would ultimately contribute to the reusability of every RBS component reachable from these cycle components.

Definition 1.2.12. (RU-preserving transformation)
A transformation δ of an input RDLT R to another RDLT R' is said to be RU-preserving if for every activity profile $S = S(1), S(2), ..., S(n_1), n_1 \in IN$, that is derivable from R, there is a corresponding activity profile $S' = S'(1), S'(2), ..., S'(n_2), n_2 \in IN$, that is derivable from R', where for every pair $(x, y) \in S(i)$ and $(y, z) \in S(i+1)$, either of the following holds:

1. $(x, y) \in S'(j)$ and $(y, z) \in S'(j+1)$,
2. there exists a path $p = x_1, x_2, ..., x_n \in R$ being represented by the arc (x_1, x_n) in R being represented by the arc (x_1, x_n) in R' where $x_{k-1} = x, x_k, x_{k+1} = z$, and
   (a) $(s, x_1) \in S'(j)$ and $(x_1, x_n) \in S'(j+1)$, for some vertex s in R',

(b) $(x_1, x_n) \in S'(j)$, for some $1 \leq j \leq n_2$, or

(c) $(x_1, x_n) \in S'(j)$ and x_n, $s \in S_{j-1}$, for some vertex s in R'.

Furthermore, we call (x_1, x_n) as the abstract arc in R' representing the path p in R.

Expanded Vertex Simplification

The process of vertex simplification results to an RDLT lacking of L-values, both in R_1 and R_2. Therefore, [7] proposes and extension of the process to produce *expanded vertex simplifiction* R'_1 and R'_2 of R by using the RDLT and its simplified vertex to retrieve the other attributes, e.g. L. The algorithm for producing R'_1 and R'_2 is outlined in 1.2.

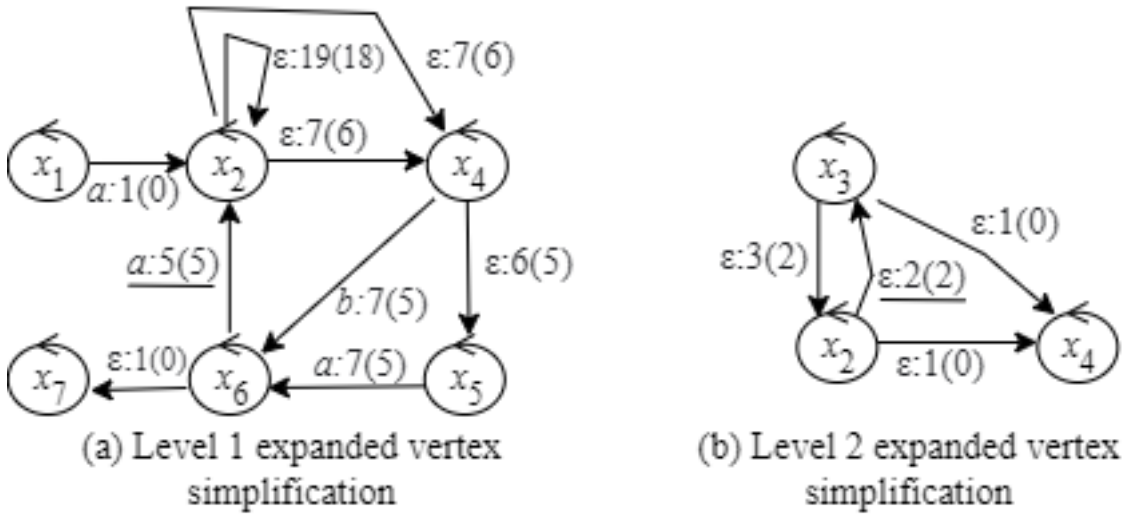

Figure 1.4: R'_1 and R'_2 expanded vertex simplification of the vertex simplified graphs in Figure 1.3

In simpler terms, the creation of a contraction path of a given vertex simplified RDLT G_1 or G_2, which can also be represented as R_1 and R_2 respectively, is the merging of a pair of vertices x and y connected by the arc (x, y) from the initial vertex through the contraction of arcs. Such a merging is only possible if the set of C-values of all arcs from x to y is a superset of the C-values of all arcs from other vertices in the subgraph towards y [7]. This continues until the subgraphs R_1 and R_2 are represented as one vertex, if possible. This method is also referred to as the graph contraction strategy.

Soundness Property in RDLT

On the soundness of RDLT, there are two formalized definitions of soundness, namely Classical and Relaxed Soundness [5, 6]. However, a recent formalization was also made for weak and easy soundness [8], the definitions of which are stated below.

Definition 1.2.13. Classical Soundness of RDLTs [6] An RDLT is of classical sound if and only if the following requirement is satisfied by each activity profile $S = \{S(1), S(2), ..., S(k)\}, 1 \leq k \leq diam(R), diam(R)$ is the diameter of R, of a set of source vertices $I \subset V$ and a final output vertex $f \in V$, where $\forall\ x \in I$ and $y \in V$, $(x, y) \in S(1)$, and $(u, f) \in S(k)$, $u \in V$:

Expanded Vertex Simplification Algorithm(EVSA):

Inputs: Let $R = (V, E, T, M)$ be an connected RDLT with an RBS, i.e. $\exists v \in V$, where $M(v) = 1$. Let $R_1 = (V_1, E_1)$ and $R_2 = (V_2, E_2)$ be the vertex simplifications of R. (The C-attributes of the arcs of R, R_1, and R_2 are denoted as C, C_1, C_2, respectively, while their L-attributes are L, L_1, and L_2, respectively.)

Outputs: Level 1 and Level 2 expanded vertex simplification $R'_1 = (V'_1, E'_1, T'_1)$ and $R'_2 = (V'_2, E'_2, T'_2)$ of R, respectively . (The C-attributes of the arcs of R'_1 and R'_2 are C'_1 and C'_2, respectively, while their L-attributes are L'_1 and L'_2, respectively.)

1. For each $v \in V_1(V_2)$(or V of R), set $v' \in V'_1(V'_2)$ to be its corresponding vertex, and if $(u, v) \in E_1(E_2)$, then its there is a corresponding arc $(u', v') \in E'_1(E'_2)$, where the C-values $C'_1(u', v')(C_2(u', v'))$ of $R'_1(R'_2)$ is set to $C(u, v)$ of $R_1(R_2)$, otherwise, none.

   This step copies the vertices and arcs, inclusive of the C-values of R_1 and R_2 to their corresponding expanded versions R'_1 and R'_2, respectively.

2. For each $(u, v) \in E$, where (u, v) is not in an RBS of R, and its corresponding arc $(u', v') \in V'_1$ of L_1, let $L'_1(u', v') = L(u, v)$.

   This step copies the L-values of R for each of its arcs to the L-values of R'_1 whenever such arc does not belong to an RBS of R.

3. For each abstract arc $(u', v') \in E'_1$ of R_1, where (u', v') represents the path $p = x_1 x_2 \dots x_n$ in R, i.e. $x_1 \in V$ and $x_n \in V$ correspond to $u' \in E'_1$ and $v' \in E'_1$, respectively, set $L'_1(u', v') = \min\limits_{i=1,\dots,n-1} \{eRU(x_i, x_{i+1})\} + 1$.

   This step sets the L-value of each abstract arc (u', v') of R'_1 to reflect the maximum number of reuse of the path that it represents in R. Note that we treat each component of the RBS as an NCA relative to the non-RBS components, $eRU(u', v')$ must be greater than its reusability, e.g. greater than the PCAs of the cycles (u', v') are involved in, hence, we add 1 to this maximum number of reuse. Additionally, note the eRU of the arcs along the path p can vary because of the reuse of such components within the RBS, albeit the number of times they are accessible through the in-bridges of their ancestor node is the same. However, the reusability of the entire path p itself is bound to the minimum of the L-values of the arcs therein. Thus, this minimum shall be the representative L-value for these arcs as reflected by their representative abstract arc.

1. **Proper termination.** For every activity profile $S = \{S(1), S(2), ..., S(k)\}, 1 \leq k \leq diam(R)$, of a set of source vertices $I \subset V$ and a final output vertex $f \in V$.

   - All arcs in the final reachability configuration $S(k)$ must be incident to sink f, i.e. for every $(x, y) \in S(k)$, $y = f$
   - If $k \leq 2$: every arc incident to a vertex y in a reachability configuration $S[i]$ has a corresponding arc incident from vertex y in a succeeding reachability configuration $S(j)$, *i.e.* for every $(x, y) \in S[i]$, there exists another arc $(y, z) \in S(j)$, for all $1 \leq i < k$, and for some j in the range $i + 1 \leq j \leq k$.

2. **Liveness.** Every arc is traversed in some activity profile, *i.e.* for every $(x, y) \in E$, there is an activity profile $S' = \{S'(1), S'(2), ..., S'(k')\}$, where $(x, y) \in S'[i], 1 \leq k \leq k'$.

Based on this definition, the first condition requires an RDLT R's each specified vertex and their subsequent vertices leading to the final vertex of R. In other words, an RDLT needs to have proper termination. The second condition requires it to have no possible occurences of deadlocks for every component in the RDLT as it requires that all arcs are to be traversed to be included in the activity profile during extraction. Figure 1.2 satisfies both conditions an is therefore classically sound.

Definition 1.2.14. Relaxed Soundness of RDLTs [5, 6, 9] An RDLT is of relaxed sound if for every sink $f \in V$ and a source $w \in R$, where w is an ancestor of f, there exists an activity profile $S = S(1), S(2), ..., S(k)$, where $1 \leq k \leq diam(R)$, where the following conditions hold:

- For every reachability configuration $S(t)$ prior to $S(k)$, if any, there should be at least one arc $(x, y) \in S(t)$ and another arc $(y, z) \in S(t')$, for a time t', where $t < t' \leq k$. This ensures that there is at least one continuous path from w to f in activity S.

- The final reachability configuration $S(k)$ is only composed of arcs that point to the sink f, *i.e.* $\forall (x, y) \in S(k)$, $y = f$.

- The set of arcs traversed at a time t is strictly contained in the set of arcs traversed at time $t + 1$, *i.e.* $\bigcup_{i=1}^{t} S[i] \subset \bigcup_{i=1}^{t+1} S[i]$. Since an arc can occur in multiple $S[i]$, the operator $\bigcup$ should be considered a multiset union operator.

- Every arc $(x, y) \in E$ is traversed in some activity profile S', *i.e.* there exists an activity profile S', where $(x, y) \in S'(t)$ for some $S'(t) \in S'$.

Relaxed soundness refers to a classically sound RDLT with the exception of the loosening of its proper termination. At least only one vertex is required to be leading to the final vertex, in contrast to a classically sound RDLT. This is given by the first, third, and fourth requirement. The liveness, however, is retained as defined in the second and third requirement, preventing deadlocks. Meaning, a relaxed soundness is live. The RDLT therefore in Figure 1.2 is relaxed sound, aside from the fact that it is classically sound in the first place.

1.3 Problem Statement

Matrix representation is used to extract activities from an RDLT, using computations involving matrix operations. The purpose of this research is to verify the weak and easy soundness of an input RDLT using these operations. Karen and Roben [4] and Asoy [3] already conducted studies involving the matrix representation of relaxed and classical soundness, respectively. Similarly, the recently formalized notion of soundness, that of weak and easy soundness in RDLT [8], provided a behavioral and structural profile upon which algorithms were proposed for the verification thereof. The problem statement of this paper is to create a matrix representation for the verification of weak and easy soundness of RDLT, based on its formalizations.

1.4 Aim of the Work

General Objectives

Currently, there are already existing literature on weak and easy soundness in the context of RDLT. However, in the case of this research, the matrix representation for the verification of these notions of soundness is to be addressed. Therefore, we shall address the following general objectives:

1. To incorporate matrix operations on the matrix representation of RDLT to verify easy and weak soundness;
2. To design a matrix representation of RDLT to verify easy and weak sound- ness; and
3. To create an algorithm for the verification of RDLT using matrix repre- sentation.

Specific Objectives

In alignment with the general objectives, the following specific objectives are identified to reach these aims:

For Weak Soundness Structural Verification

1. To establish matrix representation and operations for the verification of deadlock tolerance in both level 1 and level 2 expanded vertex simplification graphs, R_1 and R_2, respectively, of RDLT R.
2. To establish matrix representation and operations for the verification that R is deadlock-resolving.
3. To generate a list of deadlock points in R and identify escape contraction paths.
4. To establish matrix representation and operations for the verification of weakened-join safeness of R.
5. To establish matrix representation and operations to verify weakened join-safe values of every split-join pair in R.

6. To verify the loop-safeness of NCA and safeness of CA.

7. To measure the time and space complexity of the matrix-based verification of weak soundness of R.

For Easy Soundness Structural Verification

1. To establish matrix representation and operations for the verification of a contraction path existing from the source to the sink vertex.

2. To measure the time and space complexity of the matrix-based verification of easy soundness of R.

1.5 Scope and Limitations

The scope and limitations of the study are the following:

1. The study will focus on the design of matrix representation that encapsulates the structure of RDLT. It will incorporate matrix operations on the verification of the structural profiles of RDLT to determine its weak and easy soundness. The study will not delve to alternative models other than matrix representation.

2. The study will center on RDLT as the main structural model to verify notions of soundness with. Furthermore, its level 1 and level 2 expanded vertex simplified graphs will be used in the verification of weak soundness. The study will not explore other types of models beyond these graphs.

3. The study will center on the verification of weak and easy notions of soundness. Its structural profiles will be used for the verification. The study does not include other notions of soundness.

4. The study will focus on verifying deadlock tolerance in the context of RDLT, as defined in the structural profile of weak soundness. The study is limited to the definition of deadlock tolerance within this context.

5. The study aims to design and implement a matrix-based verification algorithms for weak and easy notions of soundness of an RDLT. The algorithms are only limited to weak and easy soundness and will not perform matrix operations beyond these verifications.

1.6 Significance of the Study

The study and its results will contribute to the enhanced understanding of RDLT and the verification of notions of soundness. Specifically, the study is centered on the matrix-representation of RDLT to verify weak and easy soundness, which could lead to improved methods for the analysis and modeling of complex systems. This deepens the theoretical framework of RDLT and strengthens its applicability in systems design. By establishing matrix-representation and operations for the verification of these notions of soundness, we address gaps and contirbute to the broader understanding of the field by:

1. Providing a structured framework for the verification of easy and weak soundness, which could lead to the enhancement of reliablity and accuracy of complex systems.

2. Introduce efficient deadlock detection and resolution mechanisms through matrix-based operations, leading to more robust systems.

3. Providing the groundwork for the advancement of automated verification tools, speeding up the validation process of RDLTs, in the scope of weak and easy notions of soundness.

1.7 Theoretical and Conceptual Framework

The Figure presents the conceptual framework of the research. It contains required concepts and definitions in order to proceed with the methodology and achieve the reseach objectives.

The main requirements for acheiving the main objectives of the research are the following: definition of RDLT [5], definition of a weak and easy sound RDLT based on their structural and behavioral profiles [8], RDLT Activity extraction [5] [3], Vertex Simplification, and Matrix Operations [4]. RDLT is defined in [maam malinao] and the classical notion of soundness. By the scope of this research, the weak and easy notions of soundness are to be verified, the profiles and algorithm of which are already defined by [8]. Majority of the research's objectives is focused on the matrix-based verification of the deadlock tolerance (and its subsequent properties) of the input RDLT. This methodology requires vertex simplification [5] (both level-1 and level-2) and a proposed algorithm for activity extraction [5]. Finally, the final state vector output of the matrix operations on the input RDLT will be evaluated for its validity, then its weak and easy soundness.

The implementation of this research would therefore include the reusing of the definitions of RDLT [5], weak and easy soundness profiles [8], and the algorithm for the verification thereof. The algorithm by [8] would be translated or modified to verify a matrix-based RDLT, represented by R's final state vector after activity extraction. To achieve these main objectives, specific objectives are to be achieved first, requiring the same set of concepts and algorithms. Finally, the calculation of time and space complexity of the matrix-based verification of weak and easy soundess will also be performed.

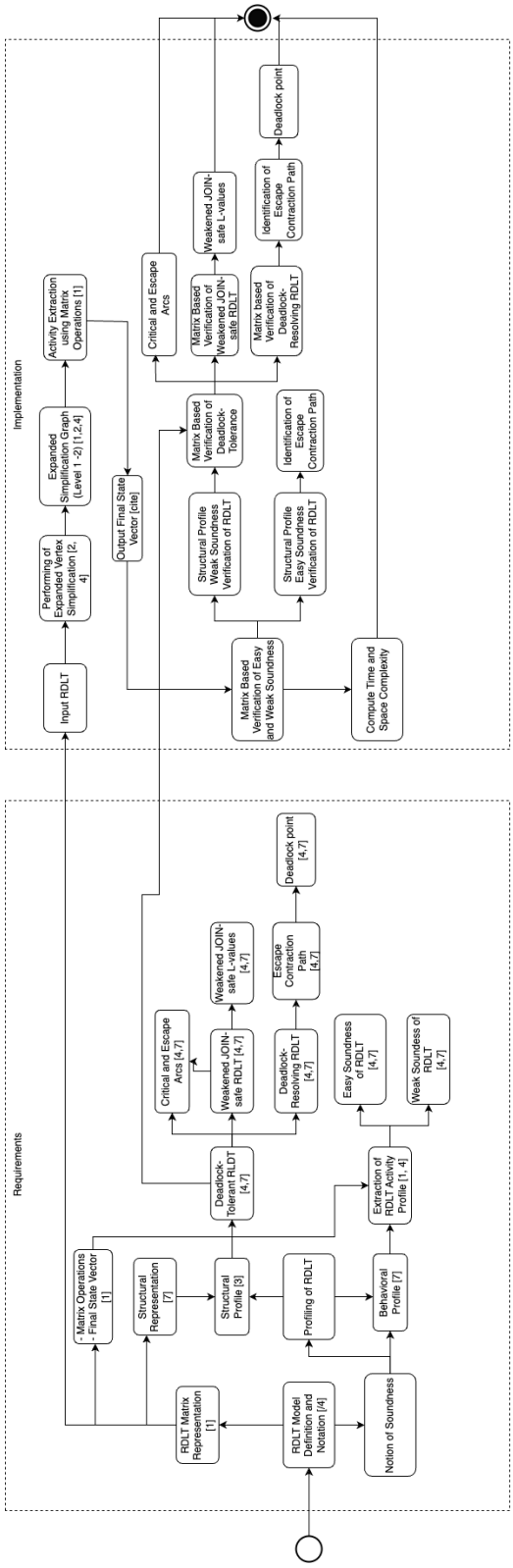

Figure 1.5: This framework shows the requirements of the research's framework, with its main components, namely: Matrix Representation, Behavioral and Structural Profile of the RDLT

Chapter 2

Review of Related Literature

On Weak and Easy Notions of Soundness in RDLTs

Recent formalizations on the definition of weak and easy soundness of RDLT and their structural and behavioral profiles allow the basis for the verification of the said properties. The following definitions are from [8] and contains the weak and easy notions of soundness, its behavioral and structural profiles, and the proposed algorithms for the verification (and the time and space complexity) thereof based on each notion's definitions. The information under this section is taken from the fundamental work of [8] unless explicitly stated otherwise.

2.1 Weak Soundness Notion of RDLTs

Definition 2.1.1 (**Weak Soundness of RDLTs**). An RDLT R is of weak soundness if and only if the following requirement is satisfied by each activity profile $S = \{S(1), S(2), ..., S(k)\}, 1 \leq k \leq diam(R), diam(R)$ is the diameter of R, of a set of source vertices $I \subset V$ and a final output vertex $f \in V$, where $\forall\ x \in I$ and $y \in V$, $(x, y) \in S(1)$, and $(u, f) \in S(k)$, $u \in V$:

1. **Proper termination.** For every activity profile $S = \{S(1), S(2), ..., S(k)\}, 1 \leq k \leq diam(R)$, of a set of source vertices $I \subset V$ and a final output vertex $f \in V$.
    - All arcs in the final reachability configuration $S(k)$ must be incident to sink f, i.e. for every $(x, y) \in S(k)$, $y = f$
    - If $k \leq 2$: every arc incident to a vertex y in a reachability configuration $S(i)$ has a corresponding arc incident from vertex y in a succeeding reachability configuration $S(j)$, i.e. for every $(x, y) \in S(i)$, there exists another arc $(y, z) \in S(j)$, for all $1 \leq i < k$, and for some j in the range $i + 1 \leq j \leq k$.

Based on this definition, an RDLT is of weak soundness if it properly terminates, where every reached vertex in each activity profile of the RDLT leads to the sink. In the context of WF-Nets, this notion of soundness in RDLT is similar in a sense that it does not require the workflow to be live. A weak sound RDLT allows for arcs to not be traversed at all.
In figure 2.1, a weak sound RDLT is shown. All the arcs leading to x_5 from x_4 can be traversed, except with the path starting from the arc (x_1, x_2), since it cannot be traversed.

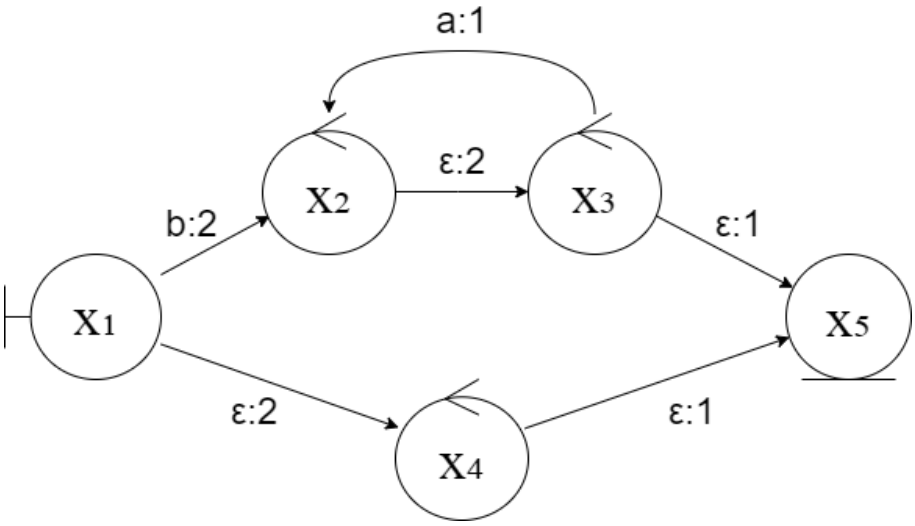

Figure 2.1: An RDLT of Weak Soundness (Image Source: [8])

2.2 Easy Soundness Notion of RDLTs

Definition 2.2.1. Self-Controlling Arc A self-controlling arc is a profile of an RDLT R where a MIX-JOIN or AND-JOIN at vertex x can never be resolved.

Figure 2.2, shown below, is an example of a self-controlling arc as defined in Definition 2.2.1. In this case, vertex b is the vertex that can never resolved due to one of the components of the AND-JOIN that violates the unconstrainedness criterion of the activity extraction algorithm. [**ADD CITATION**]

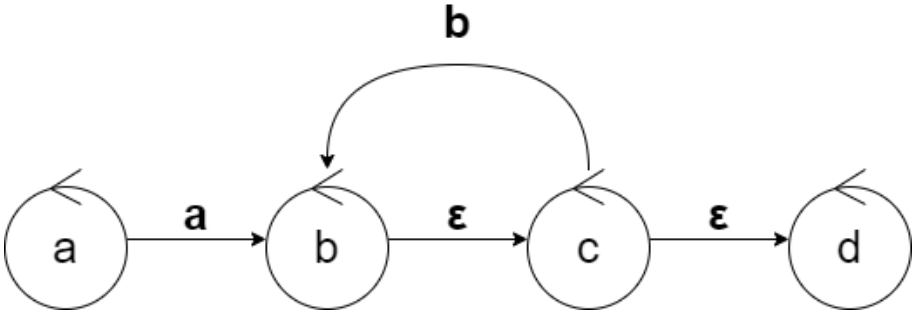

Figure 2.2: An RDLT of Easy Soundness Image Source: [8]

Definition 2.2.2. Easy Soundness of RDLTs An RDLT R is of easy soundness if and only if the following requirement is satisfied by an existing activity profile $S = \{S(1), S(2), ..., S(k)\}, 1 \leq k \leq diam(R), diam(R)$ is the diameter of R, of a set of source vertices $I \subset V$ and a final output vertex $f \in V$, where $\forall\ x \in I$ and $y \in V$, $(x, y) \in S(1)$, and $(u, f) \in S(k)$, $u \in V$:

1. **Option to complete.** There exists an activity profile $S = \{S(1), S(2), ..., S(k)\}, 1 \leq k \leq diam(R)$, of a set of source vertices $I \subset V$ and a final output vertex $f \in V$ where there exists a path P composed of $P = x_1, x_2 \cdots x_n$ with $i = 1, 2 \cdots n-2$ where $(x_i, x_{i+1}) \in S(i), (x_{i+1}, x_{i+2}) \in S(j), i < j, x_i = s$ and $x_n = f$.

As described in this definition, the only requirement for any given RDLT to be easy sound its sink to be reachable from the source. It does not require proper termination that all reached vertices must have a path that eventually leads to the sink, unlike weak soundness 2.1.1. Figure 2.4 shows an easy sound RDLT. A deadlock exist due to x_2 since there is no path from this vertex that will reach the sink x_6. However, there still exist a path from x_1 to the sink x_6 through the arcs (x_1, x_5), (x_5, x_6), fulfilling the requiremnts for an easy sound RDLT.

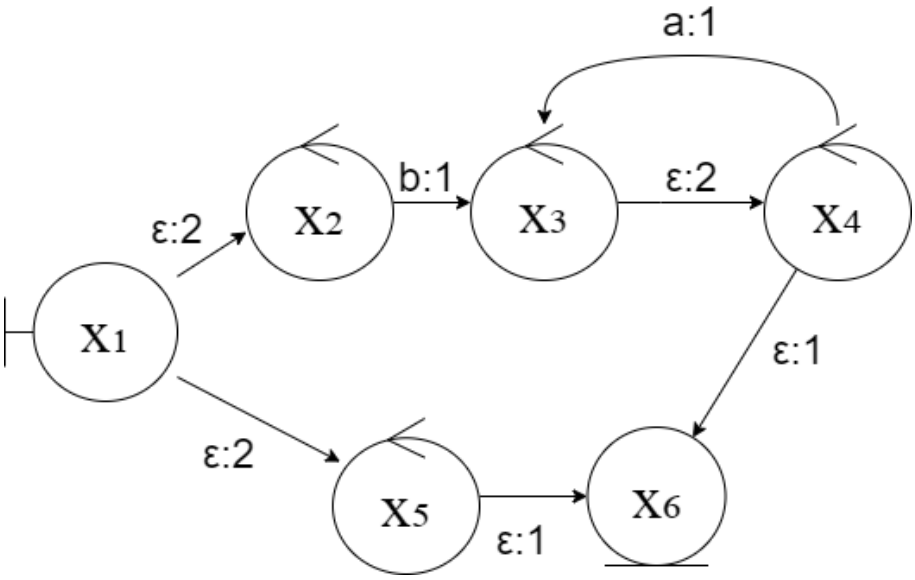

Figure 2.3: An RDLT of Easy Soundness Image Source: [8]

Remark 2.2.1. Within the context of WF-Nets, the difference between lazy and easy soundness is the number of completed cases where each case instance is represented by a token. Specifically, lazy soundness requires that there should be only one token that reaches the sink place while easy soundness allows multiple tokens in the sink place.

Currently, the activity extraction algorithm for RDLTs, as described in [5], is within the sequential context and therefore only represents one continuous case. With this, it is concluded that, sequentially, the definition of lazy and easy soundness are the same. However, lazy and easy soundness of RDLTs can be differentiated with the use of parallel activity extraction as multiple parallel cases can be visualized through this [**?**]. This difference is shown in Figure 2.4 with the sequential and parallel activity profiles. For the parallel activity, a traversal tree is utilized to visualize the various activities that are happening simultaneously. This visualization aims to structurally show and identify the parallel and non-parallel activities through observing the termination time for each branch as proposed in [**?**].

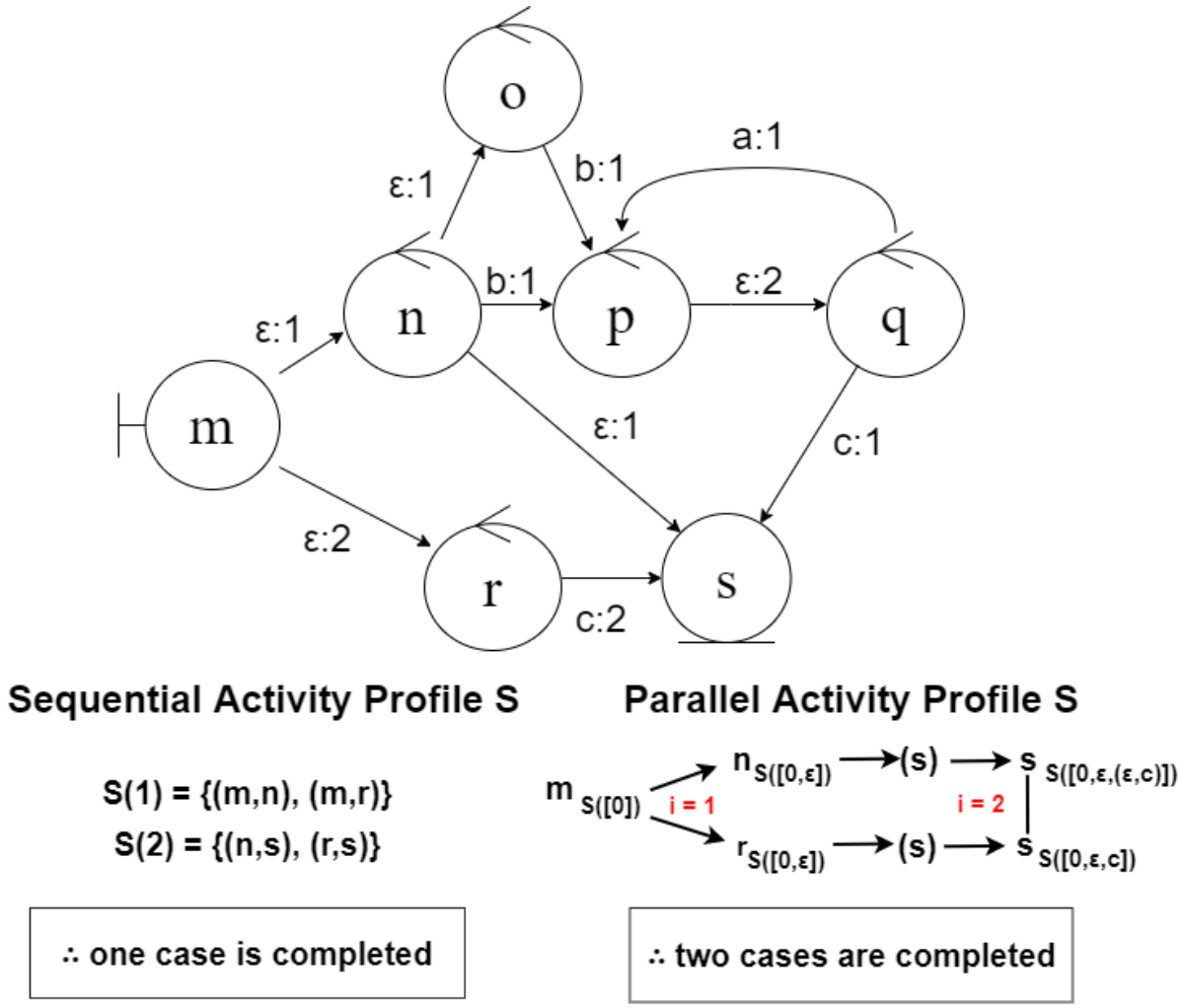

Figure 2.4: An Parallelized RDLT of Easy but not Lazy Soundness (Image Source: [8])

2.3 Behavioral Profiles of Weak and Easy Sound RDLTs

This section introduces theorems that define the behavioral profiles of both weak-sound and easy-sound RDLTs.

Theorem 2.3.1 (Implication of Weak Soundness from Classical Soundness of an RDLT R)**.** *If an RDLT R is of classical sound, then it is of weak sound.*

Proof. Let R be a classical sound RDLT.

Assume that R is not of weak sound. With this, then the following claim must hold:

1. There exists an activity profile $S = \{S(1), S(2), ..., S(k)\}, (q, o) \in S(k), q \in V, k \in \mathbb{N}$, in R, where $\exists (p, u) \in S(i), 1 \leq i < k$, such that $\nexists (u, y) \in \bigcup_{j=i+1}^{k} S(j)$. This means that S does not properly terminate through $(p, u) \in S(i)$, i.e. the child $y \in V$ of u is a pending task of an unfinished process in S.

As proved in Theorem 1 of [6], an RDLT R has been proved to terminate properly given that R is classical-sound. Through this, a classical sound RDLT satisfies the sole requirement of proper termination that an RDLT of weak soundness must have. □

Theorem 2.3.2 (Implication of Easy Soundness from Weak Soundness of an RDLT R)**.** *If an RDLT R is of weak sound, then it is of easy sound.*

Proof. Let R be a weak sound RDLT and assume it is not of easy sound. Through this, it is assumed that there is no activity profile that has a path from the source that eventually leads to the sink.

As proven in Theorem 2.4.1, the given weak-sound self-controlling RDLT was found to have proper termination even without liveness. With this, it was also proven that it had the option to complete as there must exist a path from x to the sink vertex. Through this, a weak-sound RDLT satisfies the sole requirement of the option to complete that an RDLT of easy soundness must have.

□

Corollary 2.3.3 (Implication of Easy Soundness from Classical Soundness of an RDLT R)**.** *If an RDLT R is classical sound, then it is of easy sound.*

Proof. Follows from Theorem 2.3.1 and 2.3.2. □

2.4 Structural Profiles of Weak and Easy Sound RDLTs

This section introduces theorems that define the structural profiles of both weak-sound and easy-sound RDLTs.

Theorem 2.4.1 (Proper Termination of an RDLT R of Weak Soundness)**.** *A weak sound RDLT R properly terminates even without the requirement of liveness.*

Proof. Let R be a self-controlling RDLT where deadlocks occur at (u, y). Let R also be an RDLT of weak soundness.

Due to the existence of deadlocks, R is considered to be an RDLT that is not live. To prove that a R still exhibits proper termination, we need to first establish that R is able to complete through reaching the sink vertex from the source vertex.

In this case, the activity extraction algorithm reaches x at $S(t)$ through v, i.e $(v, x) \in S(t)$. If there exists $(u, y) \in E$ and $C(u, y) \in \Sigma$, then (u, y) is constrained. Since R is of weak soundness, then there must exist a path $P = x_1, x_2, ...x - n - 2$ where $x_1 = x$ and $x_{n-2} = f$, the sink vertex.

As we now have proven the completion of R, we move on to proving its proper termination by assuming that there exists a path $P = x...f$ and $P' = x...f$ where $P \neq P'$ i.e both arrive at the same sink vertex but they overlap with each other. If the components of P and P' are in S, then they form a MIX-join or an AND-join. Moreover, since paths P and P' both complete in S, then both paths complete at the same time which proves proper termination.

□

As discussed in Chapter 1, [6] verifies the classical soundness of an RDLT if it satisfies the requirements of JOIN-safe $L-$values and $L-$safeness. [8] propsed an algorithm that takes a similar approach that uses the initial requirements and weakening them, allowing the lack of liveness in a given RDLT.

Definition 2.4.1. Weakened JOIN-safe L-values For every split-join pair of arcs (u, y), $(v, y) \in E$ in an RDLT R, (u, y) and (v, y) have weakened JOIN-safe L-values if the following conditions are met:

- For every process with $C(u, y)$ or $C(v, y) = \varepsilon$

    1. CAs are allowed as long as they are safe, i.e there exists a loop-safe non-critical EA for each CA.
    2. Branching out is allowed as resolving its JOIN is not affected by the arc that has a C-value of ε.
    3. Process interruptions are allowed regardless of the C-values of the interrupting arc.

- For every process with $C(u, y)$ or $C(v, y) = \Sigma$

    1. **One-split origin for sibling processes merging at an AND- or MIX-JOIN.** There is one common ancestor $x \in V$ for u and v such that there are is a pair of sibling processes $P_u = a_1 \dots a_n$, where $a_i = x$, $a_{n-1} = u$, $a_n = y$, $a_i \in V$, $1 < I < n - 2$, and $P_v = b_1 \dots b_m$, where $b_j = x$, $b_{m-1} = v$, $b_m = y$, $b_j \in V$, $1 < I < m - 2$. Since both paths are siblings, then they must have no arcs in common, i.e. they do not intersect except at x at y; and
    2. **No unrelated processes.** For every arc $(a, b) \in E$, if $a = x$, where x is the one common ancestor of u and v, then there is no path from a to another vertex $r \in V$ such that for some $(r, w) \in E$, $w \neq y$.
    3. **No branching out from every related process.** $\forall q_k \in Lit(P_q)$, where $q \in \{u, v\}$, $1 \leq k < |Lit(P_q)|$, $\nexists(q_{k,s}) \in E$ where $s \in V \backslash Lit(P_q)$. This is to ensure that the JOIN is resolved.
    4. **Process interruptions.** With any loss of generality, if the interrupting arc $(w, v) \in E$ has a C-value of ε, then process interruptions are allowed.
    5. **Duplicate values.**
        - If $C(u, y)$, $C(v, y) \in \Sigma$, i.e. an AND-JOIN at y, then there is no process $P = [x_1 x_2 \dots x_p]$ in R, where $x_1 = x$, $x_p = y$, such that $C(x_{p-1}, x_p) = C(u, y)$ (or $C(v, y)$).
        - Without any loss of generality, if $C(u, y) = \varepsilon$ and $C(v, y) \in \Sigma$, i.e. a MIX-JOIN at y, then any process $P = x_1 x_2 \dots x_p$ in R, where $x_1 = x$, $x_p = y$, can have $C(x_{p-1}, x_p) \in \{\varepsilon, C(v, y)\}$, i.e. duplicate C-values are allowed but no two of such arcs must have different C-values.
    6. **Equal L-values of arcs at the AND-JOIN.**
    7. **No CAs exist within the process.**

In comparison to the requirements of JOIN-safe L-values as described in Definition 1.2.9, the requirements for weakened JOIN-safe L-values is more lenient as the name suggests. Specifically, the fourth requirement is weakened such that it now allows process interruptions given that their C-value is equal to ε. Although it retains the first six requirements from the JOIN-safe L-values definition, it replaces the requirement of the loop-safeness of all related process with the requirement that there are no CAs within the process. Through these changes, this allows deadlocks to occur given that it does not occur within JOINs that has a process with a C-value with Σ. [**ADD CITATION**]

Definition 2.4.2. Weakened JOIN-safe RDLT R An RDLT R is weakened JOIN-safe if every split-join pair in R has weakened JOIN-safe L-values.

Definition 2.4.3. Deadlock Point A deadlock point is each vertex $x \in dV$ in an RDLT R where dV is a set of vertices in R and $dV \subset V$, such that $x \in dV$ is unreachable at some time step of the activity extraction algorithm.

Definition 2.4.4. Escape Contraction Path An escape contraction path is a path $P = x_1, x_2 \dots x_n$ in an RDLT R such that the following holds:

1. x_1 is a parent of a deadlock point in R, i.e. $x_1 = p$ where $(p, q) \in E$ and $q \in dV$, and

2. P is a contraction path from x_1 to x_n in R where x_n is the sink vertex of R.

Definitions 2.4.3 and 2.4.4 both describe the structures that must both exist simultaneously in a weak-sound RDLT as it assures that such an RDLT properly terminates even with the presence of deadlocks. As an example, vertex x_2 in Figure 2.1 is the deadlock point of the RDLT as it is unreachable at the first time step of the activity. The escape contraction path in this case is $P = x_1, x_4, x_5$ where x_1 is the parent of the deadlock point x_2 and x_5 is the sink vertex. [**ADD CITATION**]

Definition 2.4.5. Deadlock-Resolving RDLT R An RDLT R is deadlock-resolving if for every deadlock point $x \in V$, there exists at least one escape contraction path from a parent of x to the sink in R.

Lemma 2.4.2. *An expanded vertex simplification R_i, where $i = 1, 2$, is deadlock-tolerant if and only if every activity thereof properly terminates.*

Definition 2.4.6. Deadlock-Tolerant RDLT R An RDLT R is deadlock-tolerant if every NCA is loop-safe, every CA is safe, R is weakened JOIN-safe and deadlock-resolving.

With these definitions, the structural definition of a weak-sound RDLT is defined in Theorem 2.4.3.

Theorem 2.4.3. ***Structural Verification of the Weak Soundness of an RDLT*** *R An RDLT R is weak-sound if and only if both the level-1 and level-2 expanded vertex simplifications of R, R_1 and R_2 respectively, are deadlock-tolerant.*

With Theorem 2.4.3 as the basis, Algorithm 2 verifies the weak soundness of a given input RDLT R through the satisfaction of the requirements of deadlock-tolerance.

Algorithm 2 Weak RDLT Soundness Verification Algorithm (WRSVA)

Input: RDLT R with or without RBS
Output: True, false otherwise

```
1: Apply Expanded Vertex Simplification Algorithm [7]
2: if R contains an RBS then
3:     i = 2
4: else
5:     i = 1
6: end if
7: for every vertex simplification R_i do
8:     for every vertex v ∈ V do
9:         Store deadlock point y
10:    end for
11:    for every deadlock point y ∈ V do
12:        Determine if a non-critical loop-safe escape contraction path exists from y
13:        Determine if the split-join pair containing y has weakened JOIN-safe L-values
14:    end for
15: end for
16: if R_1 is deadlock-tolerant then
17:    if R contains an RBS then
18:        if R_2 deadlock-tolerant then
19:            return true
20:        end if
21:    end if
22:    return true
23: else
24:    return false
25: end if
```

Theorem 2.4.4. ***Time Complexity of WRSVA*** *The time complexity of verifying that an RDLT R is weak-sound is $O(c|E|^4)$.*

Proof. The algorithm is divided into three main processes: (1) expanded vertex simplification [7], (2) determining the deadlock-points, and (3) verifying deadlock tolerance. For the first process, it has a time complexity of $O(|V|^2)$ which corresponds to the maximum number of arcs R can have. Because the algorithm visits every arc of the diagram to replicate them or create abstract versions for the outputs R_1 and R_2 (if an RBS exists), the worst-case scenario for this algorithm is when the RDLT has the maximum amount of arcs, hence $O(|V|^2)$.

For the second process, it has a time complexity of $O(|V|+|E|)$, where it corresponds to the sum of the number of vertices V and arcs E in R. This process uses the depth-first or breadth-first search algorithm to solve the problem, hence $O(|V|+|E|)$.

The third process of verifying deadlock tolerance can be divided into its four requirements which has their own processes: (1) verifying loop-safeness, which takes $O(c|E|^4)$ time [6], (2) verifying existence of safe critical arcs, which takes $O(|E|^2)$ time [6], (3) verifying weakened JOIN-safeness, and (4) verifying deadlock resolution.

The verification of weakened JOIN-safeness can be further divided into its component processes and determining each of their time complexities. Specifically for every AND-

and MIX-JOIN, it concerns the processes of (1) determining the one split origin, (2) determining if there are no unrelated process, (3) determining if there are no branches, (4) determining if there are no processes interruptions with a C-value of $Sigma$, (5) determining duplicate conditions at the merge point, (6) determining the equality of the L-values, and (7) determining the non-existence of critical arcs. The first three processes take $O(|V||E|^2)$ time [6]. The fourth process takes $O(|V||E|)$ time, similar to its stricter counterpart in [6] where it avoids all process interruptions no matter the C-attribute. The fifth and sixth processes takes $O(|V||E|log|E|)$ and $O(|V||E|)$ time respectively [6]. The last process to determine the number takes $O(c|E|)$ where c is the number of cycles as it is the highest time complexity of the required steps to find so, specifically Problem 1 to 3 in Lemma 1 of [5]. Because the dominating processes in terms of time complexity are the first three processes, the time complexity of verifying weakened JOIN-safeness is $O(|V||E|^2)$.

Lastly, verifying if the RDLT R is deadlock-resolving requires $O(p(|V|+|E|))$ where p signifies the number of deadlock points since deadlock-resolving is essentially the process of graph contraction, which takes $O(|V|+|E|)$ [6], for every deadlock point in R.

With the time complexities of the sub-processes of deadlock tolerance, verifying loop-safeness has the greatest complexity out of the processes, making the time complexity of determining deadlock tolerance is $O(c|E|^4)$.

Since the time complexity of the determining deadlock tolerance is greater out of the processes as well, the weak soundness of R can be determined in $O(c|E|^4)$ time. □

Theorem 2.4.5. ***Space Complexity of WRSVA*** *The space complexity of verifying that an RDLT R is weak-sound is $O(|E|^2)$.*

Proof. As mentioned earlier, the algorithm is divided into three main processes. For the first process of EVS, it has a space complexity of $O(|V|^2)$ which corresponds to the maximum number of arcs R can have. Because the algorithm stores every arc of the diagram to replicate them or create abstract versions for the outputs R_1 and R_2 (if an RBS exists), the worst-case scenario for this algorithm is when the RDLT has the maximum amount of arcs, hence $O(|V|^2)$.

For the second process, it has a space complexity of $O(|V|+|E|)$ similar to its time complexity. It stores the vertices and arcs traversed to create the contraction path as it uses the depth-first or breadth-first search algorithm to solve the problem, hence $O(|V|+|E|)$.

The third process of verifying deadlock tolerance can be divided into its four requirements which has their own processes: (1) verifying loop-safeness, which takes $O(c|E|)$ space [6], (2) verifying existence of safe critical arcs, which takes $O(|v|+|E|)$ space [6], (3) verifying weakened JOIN-safeness, and (4) verifying deadlock resolution.

The verification of weakened JOIN-safeness can be further divided into its component processes and determining each of their space complexities. Specifically for every AND- and MIX-JOIN, it concerns the processes of (1) determining the one split origin, (2) determining if there are no unrelated process, (3) determining if there are no branches, (4) determining if there are no processes interruptions with a C-value of $Sigma$, (5) determining duplicate conditions at the merge point, (6) determining the equality of the L-values, and (7) determining the non-existence of critical arcs. The first five processes as well as the last process use $O(|E|^2)$ space [6]. However, the sixth process uses $O(|V||E|)$ space. Because the sixth process needs the least amount of space and is the sole process

with a different complexity, the space complexity of verifying weakened JOIN-safeness is $O(|E|^2)$.

Lastly, verifying if the RDLT R is deadlock-resolving requires $O(p(|V| + |E|))$ space similar to its time needed as this process is similar to the graph contraction for every deadlock point in R.

With the space complexities of the sub-processes of deadlock tolerance, verifying loop-safeness has the greatest complexity out of the processes, making the time complexity of determining deadlock tolerance is $O(|E|^2)$.

Since the space complexity of the determining deadlock tolerance is greater out of the processes as well, the weak soundness of R can be determined with $O(|E|^2)$ space. □

Shown below is Theorem 2.4.6 that describes the structural requirements required to determine the easy soundness of an RDLT.

Theorem 2.4.6. ***Structural Verification of the Easy Soundness of an RDLT*** *R An RDLT R is easy-sound if and only if the following elements exist:*

- *A contraction path P_1 in the level-1 vertex simplification R_1 of R from the initial vertex x_i to the final vertex x_f, wherein $P_1 = x_1, \ldots, x_f$*
- *A contraction path P_2 in the level-2 vertex simplification R_2 of R from the initial vertex z to the final vertex p of the RBS, wherein $P_2 = z, \ldots, p$*
- *An in-bridge (x, y) formed by a component in P_1 such that there exists a component (y, z) in P_2*
- *An out-bridge (q, p) formed by a component in P_2 such that there exists a component (p, r) in P_1*

With Theorem 2.4.6 as the basis, Algorithm 3 verifies the easy soundness of a given input RDLT R through the satisfaction of a contraction path existing from the source to the sink vertex.

Algorithm 3 Easy RDLT Soundness Verification Algorithm (ERSVA)

Input: RDLT R with or without RBS
Output: True, false otherwise

Apply Vertex Simplification Algorithm [5]
if R contains an RBS **then**
    $i = 2$
else
    $i = 1$
end if
for every vertex simplification R_i **do**
    Apply Graph Contraction Strategy [5]
end for
if R_1 has a contraction path from the source x_1 to the sink x_n **then**
    if R contains an RBS **then**
        if R_2 has a contraction path from the source z to the sink p **then**
            return true
        end if
    end if
    return true
else
    return false
end if

To illustrate this verification of easy soundness, Figure 2.5 depicts the step-by-step application of ERSVA to the easy sound RDLT shown in Figure 2.3. Since such an RDLT does not have an RBS, then it represents the level-1 vertex simplification and does not have a level-2 vertex simplification. Although this contraction does not result in one singular vertex representing all the vertices in the original figure, there exists a contraction path from the initial vertex x_1 to the final vertex x_6. This proves that the option to complete requirement is satisfied by the RDLT, thus verifying its easy soundness.

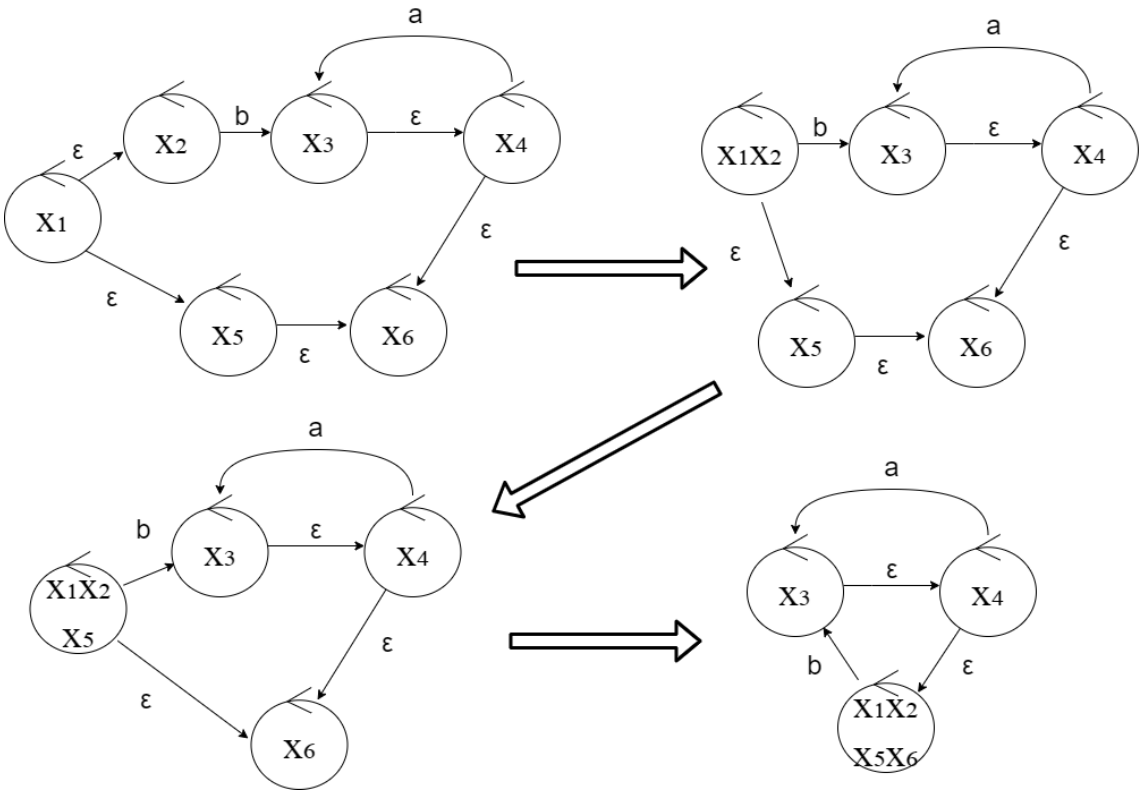

Figure 2.5: An Parallelized RDLT of Easy but not Lazy Soundness

Theorem 2.4.7. ***Time Complexity of ERSVA*** *The time complexity of verifying that an RDLT R is easy-sound is $O(|V|^2)$.*

Proof. The algorithm is divided into two main processes: (1) vertex simplification [5] and (2) graph contraction [5, 6]. For the first process, it has a time complexity of $O(|V|^2)$ which corresponds to the maximum number of arcs R can have. Because the algorithm visits every arc of the diagram to replicate them or create abstract versions for the outputs R_1 and R_2 (if an RBS exists), the worst-case scenario for this algorithm is when the RDLT has the maximum amount of arcs, hence $O(|V|^2)$.

For the second process, it has a time complexity of $O(|V|+|E|)$, where it corresponds to the sum of the number of vertices V and arcs E in R, because the process essentially determines if there exists a path from the source vertex to every vertex. Although this process is only done to make sure that there is a path from the source to the sink vertex, this complexity still applies as a worst-case scenario when there are no deadlocks within R, i.e. R is live. As proved in [6], this process uses the depth-first or breadth-first search algorithm to solve the problem, hence $O(|V| + |E|)$.

Since the time complexity of the vertex simplification is greater out of the two processes, the easy soundness of R can be determined in $O(|V|^2)$ time. □

Theorem 2.4.8. ***Space Complexity of ERSVA*** *The space complexity of verifying that an RDLT R is easy-sound is $O(|V|^2)$.*

Proof. Similar to the time complexity, the algorithm is divided into the vertex simplification and graph contraction.

For the first process, it has a space complexity of $O(|V|^2)$ as the process stores every arc within R to generate R_1 and R_2 if R has an RBS. The worst-case scenario is R having the maximum amount of arcs given a number of vertices V, hence $O(|V|^2)$.

For the second process, it has a space complexity of $O(|V| + |E|)$ [6] as the process stores the vertices and arcs to simulate the possible contraction path of both the vertex simplifications.

Since the space complexity of the vertex simplification is greater out of the two processes, the easy soundness of R can be determined using $O(|V|^2)$ space. □

On the Matrix Representation of the Verification of Soundness

Bibliography

[1] Wil Aalst. The application of petri nets to workflow management. *Journal of Circuits, Systems, and Computers*, 8:21–66, 02 1998.

[2] Willibrordus Martinus Pancratius van der Aalst. *Structural characterizations of sound workflow nets*, volume 9623 of *Computing Science Reports*. Technische Universiteit Eindhoven, Eindhoven, Netherlands, 1996.

[3] Andrei Luz B. Asoy. Automated verification of classical soundness in robustness diagrams with loop and time controls via l-safeness, 2024.

[4] Roben D. Delos Reyes and Karen Margaret D. Agnes. Matrix representation and automation of verification of soundness of robustness diagram with loop and time controls, 2018.

[5] Jasmine A. Malinao. *On Building Multidimensional Workflow Models for Complex Systems Modelling*. PhD thesis, Fakultät für Informatik (Pattern Recognition and Image Processing Group) Institute of Computer Graphics and Algorithm, Technische Universität Wien, Vienna, Austria, 2017.

[6] Jasmine A. Malinao and Richelle Ann B. Juayong. Classical soundness in robustness diagram with loop and time controls. *Philippine Journal of Science 152(6B)*, pages 2327–2342, 2023.

[7] Jasmine A. Malinao and Richelle Ann B. Juayong. Reset profiles and classical soundness in robustness diagrams with loop and time controls. *Pre-proceedings of the 12th Workshop on Computation: Theory and Practice (WCTP 2023)*, pages 371–394, 2023.

[8] Ronnie M. Ramirez II. On weak, lazy, and easy soundness in robustness diagrams with loop and time controls, 2024.

[9] Cris Niño N. Sulla and Jasmine A. Malinao. Mapping of robustness diagram with loop and time controls to petri net with considerations on soundness. In Katerina Kabassi, Phivos Mylonas, and Jaime Caro, editors, *Novel & Intelligent Digital Systems: Proceedings of the 3rd International Conference (NiDS 2023)*, pages 338–353, Cham, 2023. Springer Nature Switzerland.