(Matrix Representation for the Verification of Weak and Easy Soundness in Robustness Diagrams with Loops and Time Controls)

CMSC 199.2 - Research in Computer Science II

(Nathaniel Enrique Catalan Eulin)
(2021-08363)
B.S.C.S.

Presented to the Faculty of the
Division of Natural Sciences and Mathematics

In Partial Fulfillment of the Requirements
For the Degree of
Bachelor of Science in Computer Science

University of the Philippines
TACLOBAN COLLEGE
Tacloban City

(May) (2025)

This research problem, entitled "**(MATRIX REPRESENTATION FOR THE VERIFICATION OF WEAK AND EASY SOUNDNESS IN ROBUSTNESS DIAGRAMS WITH LOOPS AND TIME CONTROLS)**", prepared and submitted by **(NATHANIEL ENRIQUE CATALAN EULIN)**, in partial fulfillment of the requirements for the degree of **BACHELOR OF SCIENCE IN COMPUTER SCIENCE** is hereby accepted.

Research Adviser:	DR. TECHN. JASMINE A. MALINAO
Panel Members:	(TBD)
	(TBD)

Accepted as partial fulfillment of the requirements for the degree of **BACHELOR OF SCIENCE IN COMPUTER SCIENCE**.

DR. JOHN PAUL T. YUSIONG
Chair, DNSM

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background of the Study

A workflow consists of three dimensions, namely the process, resource, and case dimension [?]. 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. [?] [?].

Workflow models use 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 [?]). This includes models such as Petri nets (PN) and Robustness Diagrams with Loop and Time Controls (RDLT) [?]. These models allow workflow components to be analyzed and model properties to be verified.

The study of the workflow model RDLT 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 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. This is partly due to the separation of the represenation of the resources in a workflow to the actual processes. RDLT solves such problems, for example, by using abstractions of encapsulated activities that can be controlled by external processes (e.g., Reset-Bound Systems). In short, RDLT provides mechanims that effectively intgerate all three workflow dimensions. [?]

Soundness in Workflow Nets, eventually refered to as classical soundness, refers to the property of a workflow to be absent of livelocks, deadlocks, and other anomalies in the flow [?]. In general, to be classically sound, a model should have proper termination and liveness. There are other variations of soundness focusing on specific relaxations or modifications of the conditions proper termination and liveness, namely, relaxed, weak, easy, and lazy soundness. In the context of RDLTs, being a relatively newer workflow model, current literature shows some formalizations on notions of soundness. Classical soundness in RDLT was defined in [?] as well as relaxed soundness. Weak and easy soundness was later defined and formalized as well in [?]. Meanwhile, easy soundess in the context of RDLT is yet to be properly defined.

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 [?], they proposed a matrix representation of RDLT to verify soundness. Asoy [?] used a matrix representation of RDLT to verify and automate classical soundness. This paper aims to leverage the structural and behavioral profile of weak and easy soundness as formally defined in [?] and use matrix representations and operations to verify these notions of soundness in an RDLT.

The content of this paper is outlined as follows: Section **??** describes the basic definitions and notations of the current literature necessary to understand the topic at hand. Under this section describes brief background on Workflow Nets, Petri Nets, and its notions of soundness. It follows then the discussion on formal definitions of RDLT and its notions of soundness. Important algorithms in RDLT such as activity extraction (algorithm **??**) and vertex simplifications (algorithms **?? ??**) are described as well, including the concepts and definitions necessary. Section **??** to **??** outlines this paper's problem statement, aim, scope and limitations, significance, and the theoretical and conceptual framework of the study, respectively. Chapter **??** describes the methodology and important findings of important literatures.

1.2 Basic Definitions and Notations

Petri Nets

Petri Net [?] is a type of workflow model conceptualized by Carl Adam Petri in his disseration *Kommunikation mit Automaten* (Communication with Automata) in 1962.

This model became the basis for most modelling framework, or was inspired thereof [?]. Although having to only represent a workflow of its case and process dimensions, understanding Petri Nets and its notions of soundness (as in Workflow Nets) can also aid in our understanding of the notions of soundness in RDLT, as it is what these prior notions of soundness were built upon.

A Petri Net is a bipartite graph composed of two types of nodes, namely, places (represented by a circle) and transitions (represented by a rectangle), and an arc, which connects nodes only of different types (place to transtion, and vice versa).
A place can either be an *input* or *output* place. A place is said to be satisfied if there is at least one token inside of the place, and if it is satisfied (also called enabled), it may fire in which every output place it has recieves one token [?].

The following sections defines Petri Nets and Worklow Nets, which is a particular class of Petri Net, and follows the definitions of notions of soundness in Workflow Nets. These are important to understand as they are the basis for the formulization of the notions of soundness in the context of RDLT. The definitions in this section are mainly taken from papers [?] [?] [?] [?]with descriptions for supplemantary explanations.

Definition 1.2.1. Petri Nets and Markings [?] [?] A Petri Net is (P, T, F, M_0) which is a tuple that consists of four elements:

- a set of places P
- a set of transition T where the intersection of places and transitions must not be empty
- a set of arcs F which is a subset of the union of all the possible connections from places to transitions and vice versa.
- an initial marking $M_0 : P \to \mathbb{N}$

Figure **??** shows the components of a Petri Net.

Soundness in Workflows

A *Workflow Net* (WF-Net) is a class of Petri Net that offer capabilities of the analysis of parallel execution of activities across splits and joins of processes [**?**].

Definition 1.2.2. Strongly Connected [?]
A Petri Net is **strongly connected** if there exists a path from x to y for every pair of nodes x and y.

Figure **??** shows a simple Petri Net where every possible pair of nodes are connected, hence a **strongly connected** Petri Net.

Definition 1.2.3. Workflow Nets [?]
A Petri Net is a **Workflow Net** if such a Petri Net holds the following conditions:

- The Petri Net contains two special places which are a source place i that does not have any input places and a sink place o that does not have any output places.
- The Petri Net is strongly connected if the source place i and the sink place o are connected to a transition $t*$, as an input and output place respectively.

Given the two conditions outlined above, this would mean that the Petri Net in Figure **??** is a Workflow Net.

On the literature of Workflow Nets, all notions of soundness are already established. The notions of soundness in the context of Workflow Nets are shown in a heirarchy in Figure **??**, where arrows show a relationship of implication.

In the next sections, only classical soundness, relaxed soundness, weak, and easy soundness are discussed, as they are the ones relevant to understanding weak and easy soundness in the context of RDLT, to be discussed in further sections.

Definition 1.2.4. Classical Soundness of Workflow Nets [?]
A Workflow Net is classical sound if and only if the following requirements are satisfied:

- Option to complete: the sink place o is reachable from the source place i.
- Proper termination: the sink place o receives exactly one token and all other places must be empty.
- Liveness: every transition must be involved in at least one process specification.

This notion of soundness offeres a very strict criterion since it imposes that all process for each case are complete, leaving no pending, or cancelled task when the token reaches the sink, as explained in condition 2. [?] Figure **??** shows a classically sound workflow net. Sytems also allow the implementation of a workflow wherein at least every transition is involved in a case that is properly completed, or a token reaches the sink. This variation of soundness is less restrictive compared to that of a classically sound workflow net. The criterion is weakened to some degree, hence the following notions of soundness:

Definition 1.2.5. Relaxed Soundness of Workflow Nets [?]
A Workflow Net is of relaxed sound if and only if the following requirements are satisfied:

- Option to complete: the sink place o is reachable from the source place i.
- Proper termination (weakened): at least one case exists that completes with one token in the sink place o with other places being empty.
- Liveness: every transition must be involved in at least one process specification.

In relaxed soundness, the proper termination is weakened to at least one case in the workflow net exist where it completes with one token in the sink o while other places are empty. Therefore, it allows other cases where the workflow terminates while still having processes working in the background, represented in the model by a token inside a place, as shown in Figure **??**. Similarly, since classical soundness is implied in a relaxed sound workflow net, Figure **??** is also of relaxed sound.

Definition 1.2.6. Weak Soundness of Workflow Nets [?]
A Workflow Net is of weak sound if and only if the following requirements are satisfied:

- Option to complete: the sink place o is reachable from the source place i.
- Proper termination: the sink place o receives exactly one token and all other places must be empty.

On weak soundness in workflow nets, only two conditions are required, as defined above. The workflow should have an option to complete, and it should properly terminate. In comparison to classical soundness, it does not require liveness, that is every transition be live for every execution. An example of a weak sound workflow net is found in Figure **??**. This net is not classically sound since t_3 is a dead transition, making the workflow net not live and not fulfilling the third condition of a classically sound workflow net.

Definition 1.2.7. Easy Soundness of Workflow Nets [?]
A Workflow Net is of easy sound if and only if the following requirement is satisfied:

- Option to complete: the sink place o is reachable from the source place i.

Easy soundness in the context of workflow nets only has one condition as defined above, that is it should have an option to complete. Similarly, in Figure **??**, transition t_1 to t_2 leads to the sink o, hence has an option to complete. However, it does not satisfy the other conditions of other notions of soundness.

As a summary and ease of reference, the table below maps the requirements of soundness in the context of workflow nets **??**, obtained from [?].

Robustness Diagram with Loop and Time Controls

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

Definition 1.2.8. RDLT [?]
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 $V_{type} : V \rightarrow \{$'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,

  - $C : E \rightarrow \Sigma \cup \{\epsilon\}$ 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 $x = y$.
  - $L : E \rightarrow \mathbb{Z}$ 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.

- $M : V \rightarrow \{0, 1\}$ 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 **??** 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 **??**. 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.

Activity Extraction Algorithm

The proposed Algorithm **??** in paper [?], called *Activity Extraction Algorithm*, takes an input RDLT R with a defined start index s and goal vertex f. As shown in Algorithm **??** it returns an *activity profile* defined in Definition **??**, a set of exected vertices from the extracted activities. To understand Algorithm **??** fully, along with the definition of **activity profile**, **reachability configuration**, defined in Definition **??**, is introduced. Algorithm **??** is found at the appendices of this paper.

To perform **activity extraction** from vertex x, where $x = s$ at the start of algorithm **??**, the algorithm performs a **check**, defined in Algorithm **??** in the appendices, on x and other vertex $w \in E$ where $(x, w) \in E$. A **check**, defined in algorithm **??** at the appendices, is a subroutine in $\mathcal{A}$ where $(x, y) \in E$ is arbitrarily selected, given that the number of traversals done on (x, y) has not yet reached $L((x, y))$, which is the traversal limit of the arc, if such y exists in V. Similarly, the constraint $C((x, y))$ is checked as well, and given both $L((x, y))$ and $C((x, y))$ are satisfied, then the arc (x, y) is traversed and the next vertex in the workflow will be y. If this valid vertex y is reached, the $\mathcal{A}$ assigns time step traversal value t_i. If a previous arc or an incoming arc (u, x) prior to (x, y), the time step t_i is set to the maximum time step from all the time traversals $T((u, x))$ plus 1 from the all the incoming arcs before (x, y). Otherwise, the time traversal value is set to 1, indicating the first traversal of the arc. After the check, $\mathcal{A}$ evaluates whether every $(v, y) \in E$, $v \in V$, does not prevent the traversal on (x, y). When there are no constraints to the traversal on (x, y), the said arc is said to be an **unconstrained arc**, as defined in **??**. On the other hand, if (x, y) and (v, y) are not type-alike, (x, y) will not prevent traversal of (x, y), and (x, y) remains unconstrained with respect to all type-alike arcs with respect to the vertex y.

The following are the formal definitions of the concepts mentioned.

Definition 1.2.9. 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 1.2.10. 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 $(w, u), (y, z) \in E$.

Essentially, a **reachability configuration** is the set of arcs traversed by activity extraction at a given time step t_i, while the **activity profile** of an RDLT is the set of all reachability configurations from time step t_0 to t_d, where d is the step the workflow reached the sink.

Given these definitions, and following $\mathcal{A}$, the RDLT in Figure **??** will have the following **reachability configuration**:

$$S(1) = \{(x_1, x_2)\},$$

$$S(2) = \{(x_2, x_4)\},$$

$$S(3) = \{(x_4, x_6), (x_4, x_5), (x_5, x_6)\},$$

$$S(4) = \{(x_6, x_2)\}.$$

The **activity profile** therefore of Figure **??** is $S = \{S(1), S(2), S(3)\}$.

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 1.2.11. 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

In RDLTs, subsystems of a workflow cause an increase in the complexity of its analysis. **Vertex simplification** aims to reduce this complexity through abstraction of the components of these subsystems. In this context, the RBS is subsystem being simplified in the process of vertex simplification, thus streamlining the structure of the diagram without losing essential information necessary for analysis.

Definition 1.2.12. Vertex Simplified R [?]
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.

Figure **??** shows the vertex simplified form of the sample RDLT in Figure **??**.

Definition 1.2.13. Contraction of Arcs in RDLTs [?]
Given an RDLT $R = (V, E, T, M)$ and its vertex-simplified RDLT $G = (V', E', C')$, a contraction of $(x, y) \in E'$ of G, x, $y \in V'$, $x \neq y$, results to a multidigraph $G* = (V*, E*, C*)$ such that:

1. $V* = V' \backslash \{x, y\} \cup \{x'\}$. Here, x' is a dummy vertex representing x and y.
2. $E* = \Big\{E' \bigcup_{\forall z \in V', \exists (z,v)_i \in E', v \in \{x,y\}} \{\bigcup_{\forall i} \{(z, x')_i\}\} \bigcup_{\forall z \in V', \exists (v,z)_j \in E', v \in \{x,y\}} \{\bigcup_{\forall j} \{(x', z)_j\}\}\Big\} \backslash \Big\{ \bigcup_{\forall q \in V', \exists (q,v)_i \in E', v \in \{x,y\}} \{\bigcup_{\forall i} \{(q, v)_i\}\} \bigcup_{\forall q \in V', \exists (v,q)_j \in E', v \in \{x,y\}} \{\bigcup_{\forall j} \{(v, q)_j\}\}\Big\}$, where $z \notin \{x, y\}$, $(a, b)_i \in E'$ is the i^{th} arc from $a \in V'$ to $b \in V'$, $i = 1, 2, \ldots, n$, where n is the outdegree of a.
3. $C * ((z, x')_i) = \varepsilon$, $\forall z \in V'$ where $\exists (z, y)_i \in E'$, $i \geq 1$,
4. $C * ((x', z)_i) = C'((v, z)_i)$, $v \in \{x, y\}$, $\forall z \in V'$ where $(v, z)_i \in E'$. $i \geq 1$.

Contraction of arcs in RDLT involves the merging of two vertices into one *dummy* vertex while still preserving the properties of the arcs involved. This allows the simplification of the RDLT by reducing the amount of vertices to be analyzed.

Definition 1.2.14. Feasible Contraction [?]
A contraction of (x, y) is feasible if $\nexists (u, x) \in E'$ where $C((u, x)) \in \Sigma$, $u \in V'$ and

$$\bigcup_{\forall i} \{C'((x, y)_i)\} \cup \{\varepsilon\} \supseteq \bigcup_{\forall z \in V',\text{ where } \exists (z,y)_j \in E', j \geq 1} \{\bigcup_{\forall i} \{C'((z, y)_j)\}\},$$

where $(x, y)_i \in E'$ is the i^{th} arc from x to y, $i = 1, \ldots, n$, and n is the outdegree of x.

By this definition, the validity of contraction of vertices x and y is evaluated. It is ensured in this condition that these vertices are correct hence can be contracted into a single vertex without violating any constraints of the arcs. This ensures unerroneaous behavior with the simplified RDLT.

Definition 1.2.15. Contraction Path in RDLTs [?]
Given an RDLT $R = (V, E, T, M)$ and its vertex-simplified RDLT $G_1 = (V_1, E_1, C_1)$, a contraction path from x_1^1 to x_n in G_1 is a sequence $p = x_1^1 x_2 \dots x_n$, $n \leq |V_1|$, where a contraction is feasible on $(x_1^{i-1}, x_i) \in E_{i-1}$ in G_{i-1} resulting to $G_i = (V_i, E_i, C_i)$ for $i = 2, 3, \dots, n$, and $x_1^i \in V_i$ represents $x_1^{i-1} \in V_{i-1}$ and $x_i \in V_{i-1}$ whose arc (x_1^{i-1}, x_i) is contracted.

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 [?]. 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.*

Definition 1.2.16. Sibling Processes [?] Given a split at $x \in V$ with its processes P and P', where $P = x_1, x_2, \dots, x_n$, $P' = y_1, y_2, \dots, y_m$, P and P' are called sibling processes if $x_1 = y_1$, $x_n = y_m$, and P and P' have no arcs in common.

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 Figure **??** is shown in Figure **??**, 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.

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, along with other factors such as connectivity, and C-attributes. The following definitions discusses the L-safeness with RDLTs from [**?**].

Definition 1.2.17. Critical and Escape Arcs [**?**] A cycle $c = [x_1x_2 \ldots x_n]$ is a sequence of vertices $x_i \in V$, where $(x_i, x_{i+1}) \in E$, $i = 1, 2, \ldots, n-1$, 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.

An example of a **cycle** is x_2, x_3, where the arc from x_3 eventually leads to x_2, where x_3 can be traced, as well. The **critical arc** in this cycle is the arc with the minimum L-attribute, which is (x_2, x_3). Hence, the rest of the arcs in this cycle is a **non-critical arc**, e.g., the arc (x_3, xx_2). The **escape arc** will be the arc (x_2, x_4).

Definition 1.2.18. Loop-Safe Arcs [**?**] 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 1.2.19. Safe Critical Arcs [?] 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.

In general, escape arcs ensure that the workflow can escape from a cycle whenever the critical arc is fully used in an activity. However, having a safe critical arcs and loop-safe NCAs in R is insufficient in achieving complete soundess. In ensuring R is classically sound, an added requirement,that of $JOIN$-safe L-values, on an R having no RBS.

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 aids in ensuring the $JOIN$-safeness of an RDLT. This is one of the helpful design strategies to achieve classical soundness in JOINs, as discussed the paper [?]. This, along with defintions **??** and **??**, 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 1.2.20. JOIN-safe L-values [?] 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 $P_u = a_1a_2 \ldots a_n$, where $a_1 = x$, $a_{n-1} = u$, $a_n = y$, $a_i \in V$, $1 < I < n - 2$, and another path $P_v = b_1b_2 \ldots b_m$, 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 $P = [x_1x_2 \ldots x_p]$ 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 $P = x_1x_2 \ldots x_p$ 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 $P = x_1x_2 \ldots x_k$, $k \in \mathbb{N}$, where $x_1 = x$ and $x_k = y$, the arc $(x_1, x_{i+1}) \in E$. $i = 1, 2, \ldots, k - 1$, is loop-safe. Lastly, for an OR-JOIN merging at y, each process $P = x_1x_2 \ldots x_k$, $k \in \mathbb{N}$, of this JOIN has its arc (x_i, x_{i+1}) to have a JOIN-safe L-value, $i = 1, 2, \ldots, k-1$, 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 at least one RBS [**?**]. The information in this section is sourced from [**?**] unless explicitly mentioned otherwise.

Definition 1.2.21. 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 1.2.22. 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: $\forall p = [x_1 x_2 ... x_n] \in Cycles_{part}(R)$, 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 $d = [d_1 d_2 ... d_{m1}], e = [e_1 e_2 ... e_{m1}] \in Cycles_{part}(R)$ 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 1.2.23. (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 R lacking of L-values. Therefore, [?] proposes and extension of the process to produce *expanded vertex simplifiction* R'_1 and R'_2 with respect to 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 algorithm **??**, found in the appendices.

Soundness Property in RDLT

As explained in earlier sections, the existence of formalizations on the notions of soundness in Workflow nets served as the foundation for the formalizations of soundness on RDLT. Included in the list are classical and relaxed soundness, as formalized by [?], and weak and easy soundness by [?], all in the context of notions of soundness. The first two notions of soundness are defined below.

Definition 1.2.24. Classical Soundness of RDLTs [?] 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 $(u, f) \in S(k)$, $u \in V$:

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. All required component of a JOIN is already resolved, or checked, by the activity extraction algorithm upon termination. In other words, an RDLT needs to have proper termination for each of its activity profiles. 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 included in an activity profile during extraction, i.e., liveness of the RDLT. Figure **??** satisfies both conditions an is therefore classically sound.

Definition 1.2.25. Relaxed Soundness of RDLTs [?, ?, ?] 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 $S'(t) \in S'$.

In summary, relaxed soundness in the context of RDLT, follows the notion of relaxed soundness to that of a Workflow Net PN. It follows that the condition of proper termination of a classically sound RDLT is relaxed or weakened, that is, for an RDLT to be of relaxed soundness it is only required that there be at least one activity profile where proper termination is obeserved, while still retaining the condition of liveness of a classically sound RDLT.

1.3 Problem Statement

The verification of soundness properties of RDLT can be done through the analysis of its L-safenesss, as done in [**?**]. Matrices can then be used, as done in the papers [**?**] and [**?**], to represent the RDLT information, e.g. the vertices and arcs, the attributes, the check/traversal components, and the entire state of the RDLT during execution up until execution. The matrix representation of an RDLT is then used for the verification of classical soundness and relaxed soundness in papers [**?**] and [**?**], respectively. In the context of this study, the notions of weak and easy soundness of RDLT was just recently formalized, along with the verification strategy, in the paper of [**?**]. This study is therefore centered on the matrix representation, and using related operations, to verify weak and easy soundness of RDLT, along with the creation of matrix-based data structures, relying on the methods and formalizations of the aforementioned studies. The lack of a well-defined matrix representation tailored to the verification of weak and easy soundness of RDLT, based on recent formalizations, presents a gap in the literature that this study aims to address.

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, the general objective of the study is the following:

1. To design a matrix based representation of RDLT, and related matrix operations, for the verification of weak and easy soundness; as well as to implement the verification algorithm of [?] for the verification of weak and easy soundness using matrix representation.

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 generate a list of deadlock points in R and identify escape contraction paths [?] [?].

3. To establish matrix representation and operations to verify weakened join-safe values of every split-join [?] pair in R.

4. To establish matrix representation and operations for the verification of weakened-join safeness [?] of R.

5. To verify the loop-safeness of NCA and safeness of CA [?].

6. To establish matrix representation and operations for the verification that R is deadlock-resolving [?].

7. To formulate test cases to prove the correctness of the proposed matrix represention and operations in verifying weak soundness of RDLTs [?].

8. To prove the correctness of the proposed matrix representation and operations in verifying weak soundness of RDLTs [?].

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

For Easy Soundness Structural Verification

1. To formulate test cases to prove the correctness of the proposed matrix represention and operations in verifying easy soundness of RDLTs [?].

2. To prove the correctness of the proposed matrix representation and operations in verifying easy soundness of RDLTs [?].

3. 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, as discussed in the paper of [?] 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 achieving the main objectives of the research are the following: definition of RDLT [?], definition of a weak and easy sound RDLT based on their structural and behavioral profiles [?], RDLT Activity extraction [?] [?], Vertex Simplification, and Matrix Operations [?]. RDLT is defined in [?] 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 [?]. 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 expanded vertex simplification [?] (both level-1 and level-2) and a proposed algorithm for activity extraction [?]. 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 [?], weak and easy soundness profiles [?], and the algorithm for the verification thereof. The algorithm by [?] would be translated or modified to verify a matrix-based RDLT. To achieve these main objectives, specific objectives are to be achieved first, requiring the same set of concepts and algorithms. Finally, after formulating the test cases for verifying the correctness of the matrix-based verification of weak and easy soundness, the time and space complexity will also be performed.

Below is the concpetual framework Figure ?? which shows the concepts and components from related literature that are required to implement the objectives, displayed as yellow boxes, as well as the components for the implementations to satisfy the objectives of this study. This includes modified and novel methods, shown by orange and green boxes, respectively.