# Extended Resource Conflict Checking and Resolution Controller Design for Cross-Organization Emergency Response Processes

Matt Delengowski

***Abstract*—Discrete Event Systems (DES) models of dynamical systems which change in discrete points of time rather than continuously in time, and often asynchronously. Examples of DES are in manufacturing, logistics, healthcare, and service operations. This paper part of a final project requirement for a DES course. The requirement for the final project is to pick a subject from DES and a related IEEE paper, which was written in the last 5 years, and is from a journal that has impact factor $> 5.0$. The topic of this paper is attempting to extend the work of [1] which uses Petri Nets (PN) to model emergency response systems to optimize the allocation and dispersion of resources from first responders and down stream to hospitals. The extension is recreate the work of [1] and then attempt to make it even more efficient.**

***Index Terms*—Cross-organization emergency response processes, performance evaluation, Petri nets, resolution controller design, resource conflict checking.**

## I. Introduction

EMERGENCY response systems are highly complex with many moving parts and stages that that if performed poorly can have disastrous effects to those effected by the emergency. [1] uses a special type of Extended Petri Net to model that flow of not only information but resources between the various elements of an Emergency Response System (ERS). This work takes the invariant reduction algorithms provided in [1] for simplifying a PN and decises a program to automatically perform reductions given a simple text defintion of the PN. [1] takes the traditional 4 tuple PN definition and extends it to a total 7 tuple definition as shown in equation 1.

$$\Sigma_{\text{CE}} = (P, T, F, \alpha, \beta, W, M_0) \tag{1}$$

A subscript of $L$ denotes a logical condition for a transition to fire, a subscript of $R$ denotes a resource that must be available for a transition to fire. In other words part of the work of [1] is to have a PN that not only has conditional logic of firing off transitions but additional available resource constraints that must be met also. To begin defining the PN of equation 1, the places set $P = P_L \cup P_R$ is the union of resouce and logical places. Each resouce place has specific meaning, resusability and initial number. Additionally, resource places are differentiated as external vs internal.

| Resource Type | Resource ID | Meaning | Property | Initial Number |
|---|---|---|---|---|
| *External* | $P_{r1}$ | *ferry boat* | *Reusable* | 1 |
| | $P_{r2}$ | *public communication device* | *Reusable* | 1 |
| | $P_{r3}$ | *emergency personnel* | *Reusable* | 6 |
| | $P_{r4}$ | *media personnel and equipment* | *Reusable* | 1 |
| | $P_{r5}$ | *smoke masks* | *Consumable* | 6 |
| *Internal* | $p_{m1}$ | *emergency information* | *Consumable* | -- |
| | $p_{m2}$ | *site conditions* | *Consumable* | -- |
| | $p_{m3}$ | *medical rescue instruction* | *Consumable* | -- |
| | $p_{m4}$ | *fire rescue instruction* | *Consumable* | -- |
| | $p_{m5}$ | *search EOD instruction* | *Consumable* | -- |
| | $p_{m6}$ | *EOD search results* | *Consumable* | -- |
| | $p_{m7}$ | *fire rescue results* | *Consumable* | -- |
| | $p_{m8}$ | *medical rescue results* | *Consumable* | -- |

Fig. 1. Resource places types [1]

The transitions or *acitivites* are denoted by the set $T$ for which there are 28 elements in total. Activities can include but are not limited to *Alarm receipt*, *Rescue the wounded*, *Evacuation*, etc. $F$ denotes the token/resource flow where $F = F_L \cup F_R$. The token flow is $F_L = (P_L \times T) \cup (T \times P_L)$ and resource flow $F_R = (P_R \times T) \cup (T \times P_R)$. The weight function $W$ is modified to account for both token flow and resource movement. $W(f) = 1$ if $f \in F_L$, $W(f) = \#_{\text{req}}(t, p_r)$ is the required resources for a tranisiton to fire and $W(f) = \#_{\text{sent}}(t, p_r)$ are the resources released when a transition finishes. The intial moditions $M_0$ are initial tokens defined as $M_0(p) = 1$ if $p \in P_L$ and initial resources $M_0(p) = \#_{\text{num}}(p)$ if $p \in P_R$. Lastly, the new additionals to **??** are $\alpha$ and $\beta$ which respectively define the minimum amount of time an activitiy and the maximum amount of time an acitivty takes to execute, where $\alpha(t) < \beta(t)$.

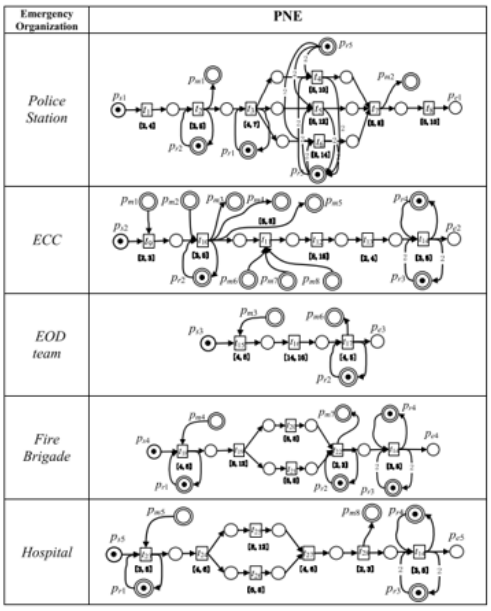


Fig. 2. Sub Petri Nets for each ERS component [1]

With the PN defintion for a single component of an ERS defined, [1] provides an example individual PNs for each component as defined in 1. Figure 2. These indivudal PNs are combined by unioning the respective sets of each component PN to form figure 3. The overall equation governing the super PN is defined in equation 2 where $i$ is a positive integer that represents a sub component of ERS.

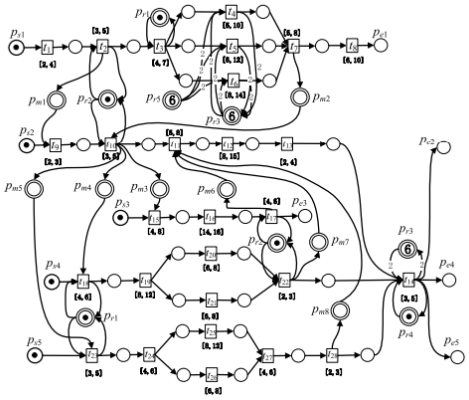

Fig. 3. Super Petri Net for ERS [1]

$$\Sigma_{\text{CE}_i} = (P_i, T_i, F_i, \alpha_i, \beta_i, W_i, M_{0_i}) \tag{2}$$

## II. Invariant Reduction

The invariant reduction as defined in [1] is a cruicial step for reducing the verbosity of the super PN in figure 3. To perform the invariant reduction [1] defines two primary forms reduction which will be referred to as sequential and fanned reduction. The sequential reduction is displayed in figure 4.a specifically which allows two sequential activities to be combined if and only if neither activity consumes or release resources then they can be safely combined. The act of combining causes the new transition to take on the sum of individual $\alpha$ and $\beta$ to create the new minimum and maximum execution times.

Figure 4.b and 4.c are special cases of figure 4.a. Respectively, they are that the first activity can consume resources and no subsequent activity can release resources or the final activity can release resources but no activity prior in the sequence can consume resources. The minimum and maximum execution times are summed uniformly.

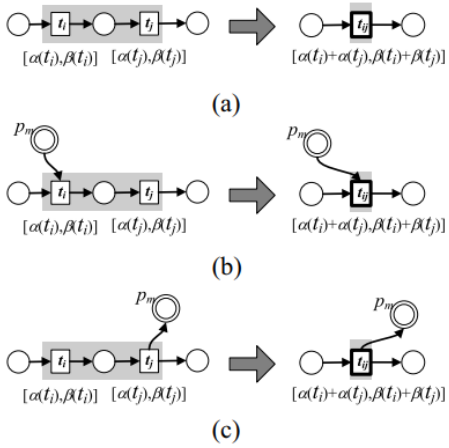


Fig. 4. Sequential reduction of activities [1]

The fanned reduction is described in figure 5 which states that two or more activties that transition out of a single activity and transition into a single activity can be combined if none of those activities consume or release resources. To combine the minimum and maximum execution times, $\alpha$ and $\beta$, take the max of each respectively, for each activity being combined.

Fig. 5. Fanned reduction of activities [1]

## III. Applied Reductions

The invariant reduction rules were applied in [1] to reduce the number of activities down from 28 to 20. This section will describe in details the individual reductions applied. Starting with activity $t_{11}$, figure 6 shows the application of reduction 1-1 to activities $t_{11}, t_{12}, t_{13}$. Reduction 1-1 can be utilized because $t_{11}$ alone consumes resources but it nor any of the subsequent activities release resources.

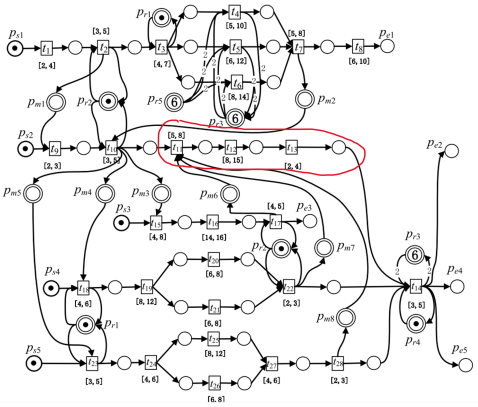

Fig. 6. Application of rule 1-1 [1]

With the first reduction applied the current number of activites has been reduced from $28 - 3 + 1 = 26$ activities.

The next application is two reductions applied iteratively. Figure 7 shows two reductions being applied sequentially. First reduction 2, the fanned reduction, is applied to $t_{25}, t_{26}$, boxed in red to create $t_{25,26}$. $t_{25,26}$ can then be reduced by through $t_{24}$ to $t_{28}$ by utilizing reduction 1-2. Of the 4 activities, $t_{24}, t_{25,26}, t_{27}, t_{28}$ no activity consumes resources and only the final activity, $t_{28}$ releases resources.

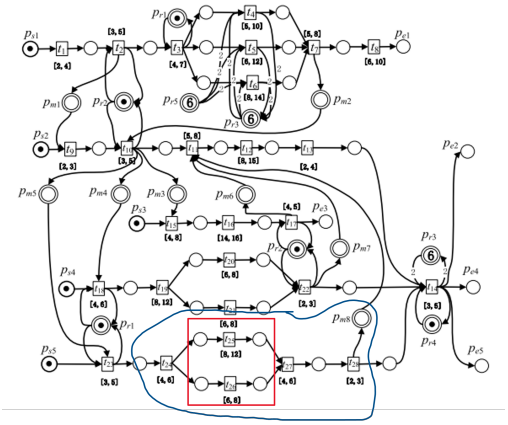

Fig. 7. Application of rule 2 - rule 1-2 [1]

The current number of activities now being reduced to $26 - 5 + 1 = 22$. The final invariant reduction application is another sequential application of multiple rules. Figure 8 shows reduction 2 being appplied and then followed up reduction 1. Reduction 2 is depicted in the red circle, which reduces the fanned activties of $t_{20}, t_{21}$ to create $t_{20,21}$. $t_{20,21}$ can then be combined with $t_{19}$ through a sequential reduction by applying reduction 1 to create $t_{19,20,21}$. After this final reduction the overall number of activites has been reduced to $22 - 3 + 1 = 20$ activities - a total reduction of 8 activities.

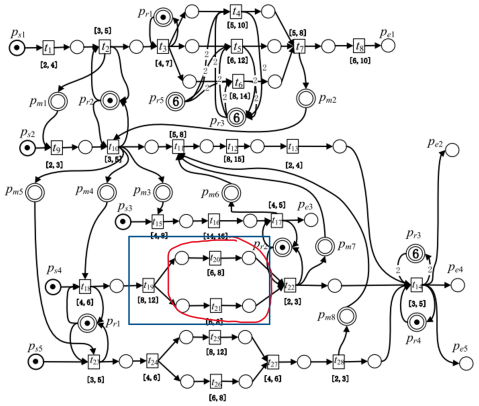

Fig. 8. Application of rule 2 - rule 1 [1]

## IV. Implementation and Discussion

The novelty of the work presented in the paper starts here. It was decided to implement a computer program to automatically apply the invariant reductions when given the text definition of resources for a PN described by 2. Figure 9 shows the table of resources definitions for the petri net. The table was transcribed to text format as a *.tsv* (tab separated values). A tsv was chosen because [1] uses definitions that include comma characters and to keep the text as close to source as possible a tab delimiter had to be chosen. The tsv in question can be found in the repository linked in the appendix in location *data/tsvs/table1.tsv*. A custom parser had to be written to transform the text format into multisets used in the table. The custom loader can be found in *dataloader.py* which contains a function *load_table1()* that performs the various Extract Load Transform (ELT) operations required to get the data into a usable format.

| Organization | Activity ID | Minimum Execution Time | Maximum Execution Time | Required Resource Set | Sent Resource Set | Pre-activities |
|---|---|---|---|---|---|---|
| *Police Station* | $t_1$ | 2 | 4 | $\emptyset$ | $\emptyset$ | $\{\emptyset\}$ |
| | $t_2$ | 3 | 5 | $\{p_{r2}\}$ | $\{p_{r2}, p_{m1}\}$ | $\{t_1\}$ |
| | $t_3$ | 4 | 7 | $\{p_{r1}\}$ | $\{p_{r1}\}$ | $\{t_2\}$ |
| | $t_4$ | 5 | 10 | $\{p_{r1}, p_{r3}\}$ | $\{p_{r3}\}$ | $\{t_3\}$ |
| | $t_5$ | 6 | 12 | $\{p_{r1}, p_{r3}\}$ | $\{p_{r3}\}$ | $\{t_3\}$ |
| | $t_6$ | 8 | 14 | $\{p_{r1}, p_{r3}\}$ | $\{p_{r3}\}$ | $\{t_3\}$ |
| | $t_7$ | 5 | 8 | $\emptyset$ | $\{p_{m2}\}$ | $\{t_4, t_5, t_6\}$ |
| | $t_8$ | 6 | 10 | $\emptyset$ | $\emptyset$ | $\{t_7\}$ |
| *Emergency Command Center* | $t_9$ | 2 | 3 | $\{p_{m1}\}$ | $\emptyset$ | $\{\emptyset\}$ |
| | $t_{10}$ | 3 | 5 | $\{p_{r2}, p_{m2}\}$ | $\{p_{r2}, p_{m3}, p_{m4}, p_{m5}\}$ | $\{t_9\}$ |
| | $t_{11}$ | 5 | 8 | $\{p_{m3}, p_{m4}, p_{m5}\}$ | $\emptyset$ | $\{t_{10}\}$ |
| | $t_{12}$ | 8 | 15 | $\emptyset$ | $\emptyset$ | $\{t_{11}\}$ |
| | $t_{13}$ | 2 | 4 | $\emptyset$ | $\emptyset$ | $\{t_{12}\}$ |
| | $t_{14}$ | 3 | 5 | $\{p_{r3}, p_{r4}\}$ | $\{p_{r3}, p_{r4}\}$ | $\{t_{13}\}$ |
| *EOD Team* | $t_{15}$ | 4 | 8 | $\{p_{m1}\}$ | $\emptyset$ | $\{\emptyset\}$ |
| | $t_{16}$ | 14 | 16 | $\emptyset$ | $\emptyset$ | $\{t_{15}\}$ |
| *Fire Brigade* | $t_{17}$ | 4 | 5 | $\{p_{r2}\}$ | $\{p_{r2}, p_{m6}\}$ | $\{t_{16}\}$ |
| | $t_{18}$ | 4 | 6 | $\{p_{r1}, p_{m6}\}$ | $\{p_{r1}\}$ | $\{\emptyset\}$ |
| | $t_{19}$ | 8 | 12 | $\emptyset$ | $\emptyset$ | $\{t_{18}\}$ |
| | $t_{20}$ | 6 | 8 | $\emptyset$ | $\emptyset$ | $\{t_{19}\}$ |
| | $t_{21}$ | 6 | 8 | $\emptyset$ | $\emptyset$ | $\{t_{20}\}$ |
| | $t_{22}$ | 2 | 3 | $\{p_{r2}\}$ | $\{p_{r2}, p_{m7}\}$ | $\{t_{20}, t_{21}\}$ |
| | $t_{24}$ | 3 | 5 | $\{p_{r3}, p_{m7}\}$ | $\{p_{r3}, p_{r4}\}$ | $\{t_{21}\}$ |
| *Hospital* | $t_{23}$ | 3 | 5 | $\{p_{r3}, p_{m8}\}$ | $\{p_{r3}\}$ | $\{\emptyset\}$ |
| | $t_{24}$ | 4 | 6 | $\emptyset$ | $\emptyset$ | $\{t_{23}\}$ |
| | $t_{25}$ | 8 | 12 | $\emptyset$ | $\emptyset$ | $\{t_{24}\}$ |
| | $t_{26}$ | 6 | 8 | $\emptyset$ | $\emptyset$ | $\{t_{25}\}$ |
| | $t_{27}$ | 4 | 6 | $\emptyset$ | $\emptyset$ | $\{t_{25}, t_{26}\}$ |
| | $t_{28}$ | 2 | 3 | $\{p_{m8}\}$ | $\{p_{r3}\}$ | $\{t_{27}\}$ |
| | $t_{14}$ | 3 | 5 | $\{p_{r3}, p_{r4}\}$ | $\{p_{r3}, p_{r4}\}$ | $\{t_{27}\}$ |
| | $T$ | $\alpha$ | $\beta$ | $\#_{req}(t, p_r)$ | $\#_{sent}(t, p_r)$ | |

Fig. 9. Resource and execution time definition for petri net. [1]

The overall ELT operation is performed by *src/objects/load_activities()* which provides a sequence of *Activity* objects given the *.tsv* file. The *Activity* class encapsulates all information given a row from the table presented in 9. To construct the *Activity* class, Python dataclasses were utilized to provide fine tuned knob turning to reduce boiler plate in coding and provide a faster implementation.

To test the implementation of the Invariant Reduction rules a dedicated testing library called *pytest* was utilzied. *pytest* was favored due to large familiarity with it and the fact that it allows easy test setup and tear down, as well as assertions to verifying the conditions of the test. Lastly, *pytest* provides very convenient debugging and reporting tools of test results to allow for fast iteration. Figures 10 and 11 show the use of pytest from command line for reporting and as a debugger, respectively.

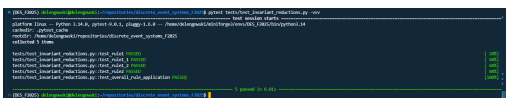

Fig. 10. Command line results of pytest with max verbosity.


Fig. 11. Debugger use of pytest

The actual implementation of the Invariant Reductions can be found in *src/invariant_reductions.py*. Each reduction variant can be found as an individual function aptly named *rule1()*, *rule1_1()*, *rule1_2()*, and *rule2()*. The application of these reductions had to be wrapped into a singular overall function call which takes place in *apply_rules()*. *apply_rules()* operates as such, apply all variants of rule 1, and determine which variant, if at all should be applied to various subsets of activities. The tie breaker logic is of various importance as situations can arise where 2 variants of rule 1 compete or compete with different, overlapping, subsets of activities.

For example, activites 11 and 12 could be combined by rule 1-1 or activities 12 and 13 could be combined by rule 1. How do you choose what goes first? Additionally, what about the case from the paper, where rule 1-1 is applied to 11, 12, and 13 at once? A special function to break ties was formulated specifically for breaking ties with variants of rule 1. Once rule 1 is applied then checks for rule 2 take place. If at any time a reduction takes place then a flag is set, indicating so, and that the iterative application of reductions should continue, as described, until no more reductons take place. Algorithm 1 shows full details in pseudo code.

The algorithm as proposed has a time complexity of $\mathcal{O}(n)$ where $n = |T|$ denotes the number of activities in the input set. Since each successful consolidation strictly reduces the size of the activity set or decreases a monotonic measure, the number of iterations is $\mathcal{O}(n)$ in the worst case.

## V. Further Work

The remaining work of [1] could be written into a computer program, particularly the algorithm for finding resource conflicts. An issue was observed when attempting to implement the resource conflict algorithm. The results could not be matched, starting with the first sub algorithm. [1] states that by applying the defintion of figure 12 the following activities should be in potential conflict $\{(t_2, t_{10}), (t_4, t_5), (t_5, t_6), (t_{18}, t_{23}), (t_{17}, t_{22})\}$. However, by the definition given, it would papear that $t_2$ and $t_{17}$ should also be in conflict as both have a required resource set of $\{p_{r2}\}$.

*Definition 6:* $\forall t_i, t_j \in T(t_i \neq t_j)$, $t_i$ and $t_j$ have *resource dependency*, denoted as $t_i \Theta t_j$, if ${}^\bullet t_i \cap {}^\bullet t_j \neq \varnothing$ and ${}^\bullet t_i \cap {}^\bullet t_j \subseteq P_R$.

Fig. 12. Definition of algorithm 1 for resource conflict detection. [1]

**Algorithm 1** Iterative Rule-Based Activity Reduction

**Require:** Activity set $T$
**Ensure:** Optimized activity set $T_{\text{opt}}$

$T_{\text{opt}} \leftarrow T$
$\mathcal{R}_1 \leftarrow \{r_1, r_{1.1}, r_{1.2}\}$
*hasReductions* $\leftarrow$ true
**while** *hasReductions* **do**
  $C_1 \leftarrow \emptyset$
  **for all** $r \in \mathcal{R}_1$ **do**
    $R \leftarrow r(T_{\text{opt}})$
    **if** $R \neq \emptyset$ **then**
      Add normalized $R$ to $C_1$
    **else**
      Add null candidate to $C_1$
    **end if**
  **end for**
  $W \leftarrow \textsc{TieBreaker}(C_1)$
  **if** $W$ contains a valid reduction **then**
    $T_{\text{opt}} \leftarrow \textsc{Consolidate}(W, T_{\text{opt}}, 1)$
  **end if**
  $R_2 \leftarrow r_2(T_{\text{opt}})$
  **if** $R_2 \neq \emptyset$ **then**
    $T_{\text{opt}} \leftarrow \textsc{Consolidate}(R_2, T_{\text{opt}}, 2)$
  **end if**
  **if** $W$ is null **and** $R_2 = \emptyset$ **then**
    *hasReductions* $\leftarrow$ false
  **end if**
**end while**
**return** $T_{\text{opt}}$

Figure 13 shows the contradicting results. There is no clear reason why $t_2$ and $t_{10}$ are in conflict but not not $t_2$ and $t_{17}$. Activities 2 and 17 both require the same resources exactly. To more specificly define what $P_R$ is, $P_R = P_{ERR} \cup P_{ECR} \cup P_{IR}$ where $P_{ERR}$ is external resuable resources, $P_{ECR}$ is external consumable resources, and $P_{IR}$ is internal resources. The resoureces are defined in figure 1. Again so activity 2, 10, and 17 all require the same external resusable resource of $p_{r2}$, so why do they all not conflict? There can only be of one of $p_{r2}$ at any time. I can only surmise that there is some issue with the paper or a misunderstanding of the text, probably the latter. It cannot even be said that perhaps only two activities can ever be in conflict at once but that would be contradicted by the fact that activty 4 conflicts with 5 and 5 conflicts with activity 6 with 4, 5, and 6 all requiring the same resources. With that said, the author does state that certain places, such as $p_{r2}$ are listed twice in the PN for clarity but perhaps that caused issues and there are slight corrections needed in the paper.

| Organization | Activity ID | Minimum Execution Time | Maximum Execution Time | Required Resource Set | Sent Resource Set | Pre-activities |
|---|---|---|---|---|---|---|
| *Police Station* | $t_1$ | 2 | 4 | $\varnothing$ | $\varnothing$ | $\{\varnothing\}$ |
| | $t_2$ | 3 | 5 | $\{p_{r2}\}$ | $\{p_{r3}, p_{m1}\}$ | $\{t_1\}$ |
| | $t_3$ | 4 | 7 | $\{p_{r3}\}$ | $\{p_{r1}\}$ | $\{t_2\}$ |
| | $t_4$ | 5 | 10 | $\{p_{r3}^2, p_{r2}^2\}$ | $\{p_{r3}^2\}$ | $\{t_3\}$ |
| | $t_5$ | 6 | 12 | $\{p_{r3}^2, p_{r2}^2\}$ | $\{p_{r3}^2\}$ | $\{t_3\}$ |
| | $t_6$ | 8 | 14 | $\{p_{r3}^2, p_{r2}^2\}$ | $\{p_{r3}^2\}$ | $\{t_3\}$ |
| | $t_7$ | 5 | 8 | $\varnothing$ | $\{p_{m2}\}$ | $\{t_4, t_5, t_6\}$ |
| | $t_8$ | 6 | 10 | $\varnothing$ | $\varnothing$ | $\{t_7\}$ |
| *Emergency Command Center* | $t_9$ | 2 | 3 | $\{p_{m1}\}$ | $\varnothing$ | $\{\varnothing\}$ |
| | $t_{10}$ | 3 | 5 | $\{p_{r2}, p_{m1}\}$ | $\{p_{r2}, p_{m1}, p_{m3}, p_{m4}\}$ | $\{t_9\}$ |
| | $t_{11}$ | 5 | 8 | $\{p_{m4}, p_{m3}, p_{m4}\}$ | $\varnothing$ | $\{t_{10}\}$ |
| | $t_{12}$ | 8 | 15 | $\varnothing$ | $\varnothing$ | $\{t_{11}\}$ |
| | $t_{13}$ | 2 | 4 | $\varnothing$ | $\varnothing$ | $\{t_{12}\}$ |
| | $t_{14}$ | 3 | 5 | $\{p_{r3}^2, p_{r4}\}$ | $\{p_{r3}^2, p_{r4}\}$ | $\{t_{13}\}$ |
| *EOD Team* | $t_{15}$ | 4 | 8 | $\{p_{m3}\}$ | $\varnothing$ | $\{\varnothing\}$ |
| | $t_{16}$ | 14 | 16 | $\varnothing$ | $\varnothing$ | $\{t_{15}\}$ |
| | $t_{17}$ | 4 | 5 | $\{p_{r2}\}$ | $\{p_{r2}, p_{m1}\}$ | $\{t_{14}\}$ |
| *Fire Brigade* | $t_{18}$ | 4 | 6 | $\{p_{r1}, p_{m4}\}$ | $\{p_{r1}\}$ | $\{\varnothing\}$ |
| | $t_{19}$ | 8 | 12 | $\varnothing$ | $\varnothing$ | $\{t_{18}\}$ |
| | $t_{20}$ | 6 | 8 | $\varnothing$ | $\varnothing$ | $\{t_{19}\}$ |
| | $t_{21}$ | 6 | 8 | $\varnothing$ | $\varnothing$ | $\{t_{19}\}$ |
| | $t_{22}$ | 2 | 3 | $\{p_{r3}\}$ | $\{p_{r3}, p_{m3}\}$ | $\{p_{20}, t_{21}\}$ |
| | $t_{14}$ | 3 | 5 | $\{p_{r3}^2, p_{r4}\}$ | $\{p_{r3}^2, p_{r4}\}$ | $\{t_{13}\}$ |
| *Hospital* | $t_{23}$ | 3 | 5 | $\{p_{r1}, p_{m1}\}$ | $\{p_{r1}\}$ | $\{\varnothing\}$ |
| | $t_{24}$ | 4 | 6 | $\varnothing$ | $\varnothing$ | $\{t_{23}\}$ |
| | $t_{25}$ | 8 | 12 | $\varnothing$ | $\varnothing$ | $\{t_{24}\}$ |
| | $t_{26}$ | 6 | 8 | $\varnothing$ | $\varnothing$ | $\{t_{25}\}$ |
| | $t_{27}$ | 4 | 6 | $\varnothing$ | $\varnothing$ | $\{t_{25}, t_{26}\}$ |
| | $t_{28}$ | 2 | 3 | $\varnothing$ | $\{p_{m4}\}$ | $\{t_{27}\}$ |
| | $t_{14}$ | 3 | 5 | $\{p_{r3}^2, p_{r4}\}$ | $\{p_{r3}^2, p_{r4}\}$ | $\{t_{13}\}$ |

Fig. 13. Conflicting results of algorithm 1. Why are activities 2 and 17 not in conflict? [1]

## VI. Conclusion

In conclusion an algorithm was successfully devised to implement invariant reductions for a ERS petri net, given the text defintion' of its resources places. In short, this can provide use for large petri nets to provide faster reduction in complexity of the PN and allow quicker time to conflict indentification and resolution. Further work could be performed to write computer programs to implement the conflict identification.

## Appendix
## Source Code

The source code for conducting experiments and this paper can be found at a public github repository. https://github.com/Delengowski/discrete_event_systems_F2025

## References

[1] Qingtian Zeng et al. "Resource Conflict Checking and Resolution Controller Design for Cross-Organization Emergency Response Processes". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 50.10 (Oct. 2020), pp. 3685–3700. ISSN: 2168-2232. DOI: 10.1109/TSMC.2019.2906335.