Cycl○n – A Tool for Determining Stop-Transitions of Petri Nets

Jörg Desel, Marc Finthammer(✉), and Andrea Frank

FernUniversität in Hagen, Hagen, Germany
{joerg.desel,marc.finthammer,andrea.frank}@fernuni-hagen.de

Abstract. This paper introduces the tool Cycl○n. The core functionality of Cycl○n is to determine for a transition t of an unbounded Petri net whether or not t stops the net. A transition t stops the net (and is called a stop-transition) if each reachable marking of the net enables only finite occurrence sequences without occurrences of t. Cycl○n provides a graphical user interface which also illustrates the graph structures leading to the computed result. This way, results are explained in a comprehensible manner, and the user gets a visual explanation of the cyclic behavior of the net causing these results.

Keywords: Unbounded Petri nets · Coverability graphs · Termination

1 Introduction

The core functionality of the tool Cycl○n is to provide a practical solution to the following problem, originally introduced in [1]: Assuming a Petri net[1] and a transition t of this net, can we eventually stop the behavior of the net by forbidding occurrences of t in occurrence sequences enabled at an arbitrary reachable marking? Or, equivalently: Does no reachable marking of the net enable an infinite occurrence sequence without occurrences of t? If this is the case, then we say that *transition t stops the net* or that *t is a stop-transition*. Consequently, if t does not stop the net, then some reachable marking enables an infinite occurrence sequence without occurrences of t.

As thoroughly explained in [1], determining if a transition t stops an *unbounded* net is a difficult task, since just checking a coverability graph of the net for cycles[2] does not provide a sufficient criterion for unbounded nets[3]. However, it is shown in [1] that t stops the net if and only if there is no *non-decreasing closed path* of the coverability graph without an arc labeled by t. A closed path of the coverability graph is called non-decreasing, if, for every place

[1] Cycl○n handles place/transition Petri nets without arc weights, capacity restrictions or inhibitor arcs (in accordance with the nets considered in [1]).

[2] A closed path is a *cycle* if no two distinct arcs of the path start at the same node.

[3] As also shown in [1], considering stop-transitions in *bounded* nets is a much simpler task, since indeed it merely requires to check cycles in the reachability graph.

R. Janicki et al. (Eds.): PETRI NETS 2020, LNCS 12152, pp. 392–402, 2020.
https://doi.org/10.1007/978-3-030-51831-8_20

of the net, at least as many tokens are added as are removed by the transitions at the arcs of the path, and hence the effect of these transitions is non-negative for each place. An algorithm has been developed in [1] which – very roughly speaking – identifies and processes non-decreasing closed paths in a coverability graph of the net and that way determines stop-transitions. The algorithm involves constructing a homogeneous system of linear inequalities over the arcs of the coverability graph, computing a basis of solutions and performing a divide-and-conquer approach on the solutions to finally determine if a transition t stops the net.

Cycl○n implements this algorithm. It provides a graphical user interface and performs all necessary computations in the background, making it easy for the user to examine if a selected transition stops the net. Moreover, Cycl○n does not only deliver a pure yes-or-no result, but also features a graphical representation of the involved graph structures to illustrate and explain the computed result. That way, Cycl○n allows the user to comprehend the cyclic behavior of the net leading to the result. Furthermore, Cycl○n offers additional features, e.g. considering several transitions simultaneously, automatically investigating each transition of the net in a batch-processing manner, focusing on a certain subset of the net, and visualizing cyclic behavior in a more general way (i.e. without a priori considering a certain transition).

In this contribution, we will illustrate all functionalities of Cycl○n in detail by means of examples. Note that some of these functionalities are in close connection to the theoretical background explained in [1]. Hence, we encourage the reader to see [1] for a deeper understanding of some of the concepts, which we can only outline here very briefly.

2 User Interface and Functionalities

Figure 1 shows the graphical user interface of the Cycl○n tool. Cycl○n supports Petri net models specified in the PNML file format. After opening a PNML file, a graphical representation of the net is displayed in the upper left part of the GUI and a coverability graph (CG) of the net is constructed[4] and displayed in the upper right part. Each ω-marking m_ω in CG is represented as a tuple $(m_\omega(p_1), \ldots, m_\omega(p_n))$, i.e., each such tuple is implicitly based on the given order of places. Each arc of CG is labeled by $[\mathtt{a_i}]$ $\mathtt{t}$ with $\mathtt{a_i}$ being a distinct arc identifier and $\mathtt{t}$ corresponding to the respective transition.

The output area in the lower left part informs about the results of the computation and the lower right part holds a selectable list of computed basis solutions (which we will explain in Sect. 2.2). The GUI has a toolbar which provides buttons for all functionalities of the Cycl○n tool. Above the toolbar is another array of buttons which provides quick access to several preconfigured examples (i.e. PNML files with preselected transitions).

[4] The coverability graph is computed by a non-deterministic algorithm; consequently, the structure and arc numbering of the particular coverability graph may change with each computation.

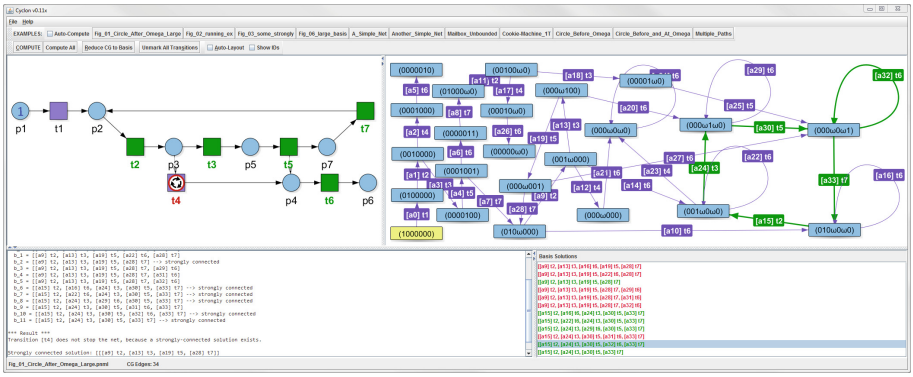

Fig. 1. Graphical user interface of the Cyclon tool

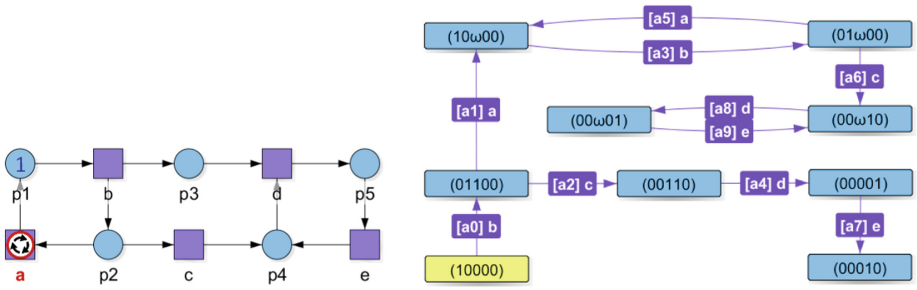

Fig. 2. An unbounded net with transition a selected as a forbidden transition and a coverability graph of the net

2.1 Core Functionalities

Figure 2 shows an unbounded[5] Petri net (which we will use as a running example throughout this section) and a coverability graph CG of this net. Note that simply taking a closer look at CG does not help in determining which transitions stops the net: Obviously, the coverability graph contains two[6] closed paths (even cycles), but we have to search for non-decreasing closed paths in order to determine stop-transitions. In the following, we will demonstrate how to employ the Cyclon tool for this task.

To examine whether or not a certain transition stops the net, e.g. transition a, the user selects the respective transition via mouse click (indicted by the

[5] Note that Cyclon also handles bounded Petri nets, but we focus on the much more challenging and interesting case of unbounded nets in this paper.

[6] Precisely, there are infinitely many closed paths, since e.g. not only $a5, a3$ is a closed path, but also $a5, a3, a5, a3$, and so on.

red-white sign[7] on transition a in Fig. 2). Roughly speaking, such a selected transition is considered to be forbidden to occur infinity often, therefore we also refer to a selected transition as a *forbidden* transition. Pressing the "Compute" button starts the algorithmic computation to determine if the selected transition a is a stop-transition. The following result is presented in the output area:

```
Forbidden Transition: T = [a]
*** Result ***
Transition [a] stops the net.
```

That is, if we forbid transition a to occur infinitely often, then the net eventually terminates. This result follows directly from the algorithmic computation which has shown that there is no non-decreasing closed path in CG without an arc labeled by transition a. Expressed the other way round: Every non-decreasing closed path in CG contains an arc labeled by transition a.

Next, we select transition e instead. Performing the computation yields:

```
Forbidden Transition: T = [e]
Basis Solutions:
  b_0 = [[a3] b, [a5] a] --> strongly connected
*** Result ***
Transition [e] does not stop the net,
because a strongly-connected solution exists: [[[a3] b, [a5] a]]
```

That is, even if we forbid transition e to occur infinitely often, the net does not terminate necessarily, since there exists an infinite occurrence sequence which does not rely on transition e. The output also gives reasons for the result by presenting a particular *strongly connected solution*, which – in the context of [1] – corresponds directly to a non-decreasing closed path without an arc labeled by transition e. The non-decreasing closed path indicated by the output `[[a3] b, [a5] a]` is also highlighted green in CG. So the green arcs `a3` and `a5` represent a non-decreasing closed path which has no arc labeled by transition e. Consequently, this non-decreasing closed path leads to an infinite occurrence sequence $b, a, b, a, \ldots$ without transition e and therefore transition e does not stop the net. The involved transitions a and b are also highlighted green in the graphical representation of the net. The graphical representation of CG makes it easy to see that the infinite occurrence sequence is enabled at marking (`10ω00`) and that the (finite) sequence b, a leads from the initial marking to that particular marking.

By clicking on a marking in CG, this particular marking becomes highlighted and is displayed in the graphical representation of the net, i.e. the respective number of tokens is displayed on each place of the net. This allows to click several markings in CG (e.g. the markings involved in a non-decreasing closed path) one by one and retrace the corresponding markings in the net, in this way e.g. studying the cyclic behavior of the net.

Cycl○n also allows to select multiple transitions at once in order to examine if forbidding these transitions simultaneously stops the net. For example, selecting transitions c, d, and e gives the result:

[7] The red-white "traffic-circle-prohibition-sign" shall give an intuitive hint that this transition is not "absolutely" forbidden to occur, but it is forbidden to occur in "infinite cyclic activities" (very roughly speaking).

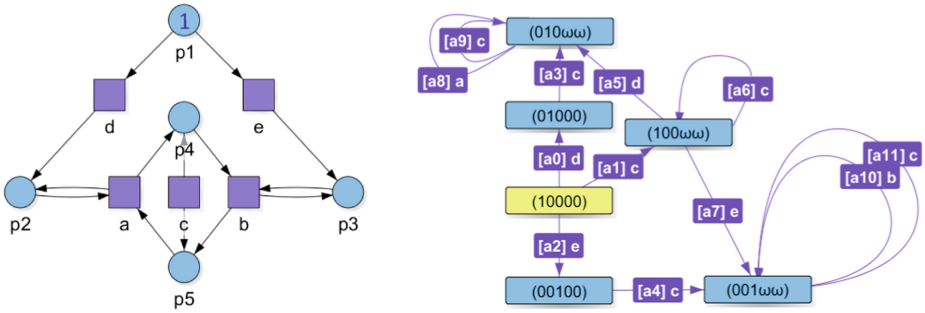

Fig. 3. An unbounded net which leads to several basis solutions; a coverability graph of the net

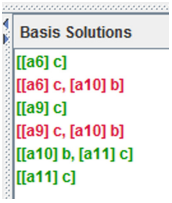

Fig. 4. Multi-selectable list of basis solutions in the GUI (green: strongly connected, red: not strongly connected) resulting from computation for transition a (Color figure online)

```
Forbidden Transitions: T = [c, d, e]
Basis Solutions:
  b_0 = [[a3] b, [a5] a] --> strongly connected
*** Result ***
Transitions [c, d, e] do not stop the net,
because a strongly-connected solution exists: [[[a3] b, [a5] a]]
```

Considering the previous result, this does not surprise, since the already known strongly connected solution `[[a3] b, [a5] a]` (non-decreasing closed path, respectively) does not contain any of these transitions.

Furthermore, Cyclon features some sort of batch processing to check for each transition of the net automatically if it stops the net. Starting this batch processing via the "Compute All" button results in the following output:

```
*** Results for All Transitions ***
Transition [a] stops the net? true
Transition [b] stops the net? true
Transition [c] stops the net? false   strongly connected solution: [[[a3] b, [a5] a]]
Transition [d] stops the net? false   strongly connected solution: [[[a3] b, [a5] a]]
Transition [e] stops the net? false   strongly connected solution: [[[a3] b, [a5] a]]
```

2.2 Visualization of Basis Solutions

Now we consider the net and a coverability graph CG of this net depicted in Fig. 3. Selecting transition c yields:

```
Forbidden Transition: T = [c]
Basis Solutions:
  b_0 = [[a8] a, [a10] b]
*** Result ***
Transition [c] stops the net.
```

Note that the algorithm has determined a basis solution. However this basis solution is not strongly connected, i.e. the respective arcs `a8` and `a10` in CG do not constitute a strongly connected subgraph of CG; but being strongly connected is a necessary property of a closed path. Consequently, since no strongly connected solution exists, there is no non-decreasing closed path either and therefore transition c stops the net.

By performing the computation for transition a instead, we get a more comprehensive result:

```
Forbidden Transition: T = [a]
Basis Solutions:
  b_0 = [[a6]  c]          --> strongly connected
  b_1 = [[a6]  c, [a10] b]
  b_2 = [[a9]  c]          --> strongly connected
  b_3 = [[a9]  c, [a10] b]
  b_4 = [[a10] b, [a11] c] --> strongly connected
  b_5 = [[a11] c]          --> strongly connected
*** Result ***
Transition [a] does not stop the net,
because a strongly-connected solution exists: [[[a6] c]]
```

As we have already seen, the result that a transition stops the net is always justified by an appropriate strongly connected solution. However, this time the output also informs us about six basis solutions in total, four of them being strongly connected. All basis solutions are also listed in the lower-right area of the GUI (see Fig. 4). The color of each basis solution indicates whether it is strongly connected (green) or not (red). Selecting a solution from the list highlights the appropriate arcs in CG as well as the corresponding transitions in the net in the respective color (see Fig. 5a).

Moreover, multiple basis solutions can be selected at once and Cycl○n determines on-the-fly for that particular set of basis solutions (i.e. for the induced subgraph) whether it is strongly connected or not. To indicate the result, all involved arcs in CG (and transitions in the net) are highlighted in green or red, respectively (see Fig. 5b). That way, the user can try out different combinations of basis solutions and receives a visual information of which arcs (and transitions) are involved and whether they are strongly connected or not.

2.3 Focusing on a Part of the Net

Next, we consider the net and coverability graph CG depicted in Fig. 6. Cycl○n also supports to determine the answer to a more general version of our original question, namely: Does a transition t stop all transitions from *a certain part* of the net? That is, with U being a subset of transitions of the net, we ask: Does

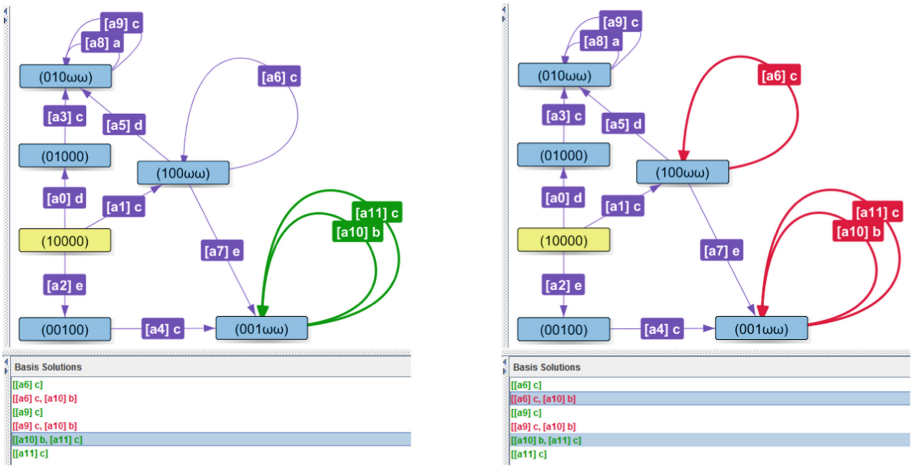

Fig. 5. a) A strongly connected basis solution is selected in the list, and the corresponding arcs in the coverability graph are highlighted green **b)** Two basis solutions (one strongly connected, the other not strongly connected) are selected in the list, and the corresponding arcs in the coverability graph are highlighted red, since together these basis solutions are not strongly connected. (Color figure online)

transition t stop all transitions in U? So we ask if there is no non-decreasing closed path which does not contain t but contains some transition from U. Note that if U contains all transitions of the net, then this generalized question coincides with our original question.

To define such a subset U of transitions, the user selects the respective transitions by holding the Shift-key and clicking them, that way labeling these transitions with yellow "or" lettering. In Fig. 6, we have selected transitions a and b that way, and additionally selected transition c as a forbidden transition. The computation gives the following result (note that we have abbreviated the output by leaving out some not strongly connected basis solutions; see Fig. 7a for the whole set of basis solutions):

```
Forbidden Transition: T = [c]
Part of the net: U = [a, b]
Basis Solutions:
  b_0  = [[a30] f, [a39] g] --> strongly connected
  b_1  = [[a32] a, [a47] b]
  ...
  b_8  = [[a58] a, [a67] b]
  ...
  b_11 = [[a59] f, [a68] g] --> strongly connected
  ...
  b_17 = [[a65] a, [a71] b]
*** Result ***
Transition [c] does not stop all transitions in U = [a, b], because a
strongly-connected solution exists which contains some of these transitions.
Strongly connected solution: [[[a58] a, [a67] b], [[a59] f, [a68] g]]
```

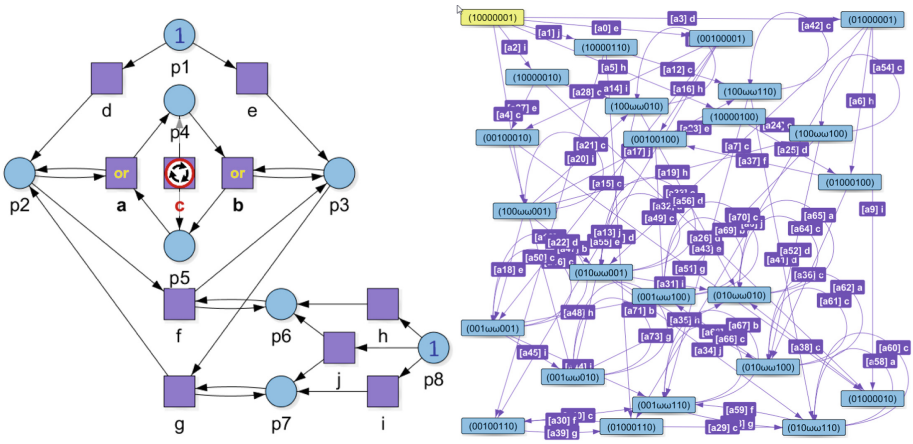

Fig. 6. An unbounded net with transition c selected as forbidden, and transitions a and b selected as the part of the net to focus on; a coverability graph of the net consisting of 74 arcs

Before discussing the above result, we quickly point out another feature of Cycl○n which is very useful at this point: After performing a computation, Cycl○n allows to reduce CG to the subgraph which is constituted by the determined set of basis solutions (see Fig. 7a). Pushing the "Reduce CG to Basis" button generates the subgraph shown in Fig. 7b, which offers much more clarity, by leaving out arcs and markings not relevant with respect to the set of basis solutions.

Next, we take a closer look at the above result: Due to the more generalized version of our question, finding a strongly connected solution which does not contain transition c is not sufficient, but we also require that transition a or b are contained in such a solution. Therefore, the divide-and-conquer part of the performed algorithm has to determine if there exists a combination of basis solutions which satisfies both requirements. The determined solution `[[[a58] a, [a67] b], [[a59] f, [a68] g]]`, which is a combination of the strongly connected basis solution `[[a59] f, [a68] g]` and the not strongly connected basis solution `[[[a58] a, [a67] b]` (cf. the selected solutions in Fig. 7a), is strongly connected itself (indicated by the green arcs in Fig. 7b). Hence, it corresponds to a non-decreasing closed path not containing c but containing some transition from $U = \{a, b\}$ (it even contains both transitions).

2.4 Analyzing Basis Solutions Beyond Stop-Transitions

Although Cycl○n has been developed for determining stop-transitions, it offers another useful feature to investigate the cyclic behavior of a net in a more general way. To illustrate this, we once again come back to the net from Fig. 3. However, this time we perform the computation without selecting any transition at all,

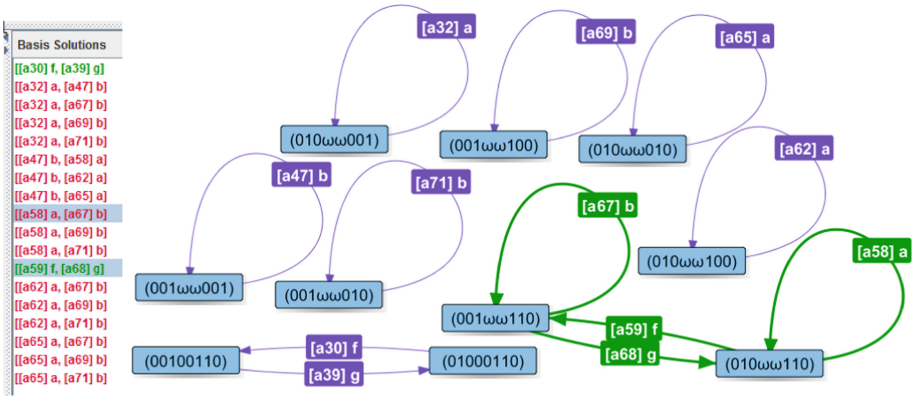

Fig. 7. a) List of basis solutions for the setting from Fig. 6 with a "green" and a "red" basis solution selected which together are strongly connected **b)** Subgraph of the coverability graph from Fig. 6 which is reduced to the arcs contained in the basis solutions (Color figure online)

i.e. we do not forbid any transition, and get the following result (again, we abbreviated the result; see Fig. 8a for the whole set of basis solutions):

```
Forbidden Transitions: T = []
Basis Solutions:
  b_0 = [[a6]  c]          --> strongly connected
  b_1 = [[a6]  c, [a8] a]
  ...
  b_8 = [[a10] b, [a11] c] --> strongly connected
  b_9 = [[a11] c]          --> strongly connected
*** Result ***
Transitions [] do not stop the net,
because a strongly-connected solution exists: [[[a6] c]]
```

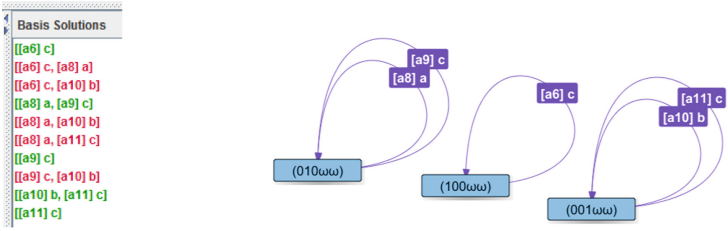

Fig. 8. a) List of basis solutions for the net from Fig. 3, if performing a "dry run" for this net (i.e. performing the computation without any forbidden transition) **b)** Subgraph of the coverability graph from Fig. 3 which is reduced to the arcs contained in the basis solutions

If we have an unbounded net and perform such a "dry run" (i.e. we do not forbid any transition), then there always exists a strongly connected solution

corresponding to a non-decreasing closed path in CG. By performing the computation without forbidding any transition, we get an "unrestricted" set of basis solutions. We can use the functionalities of Cycl○n to take a closer look at these basis solutions, e.g. by investigating the involved arcs and transitions, and by checking if some combination of basis solutions is strongly connected. In this way we can already get some intuitive insights to the cyclic behavior of a net[8] in a practical way.

For example, a closer look at the determined basis solutions together with the reduced CG (see Fig. 8) makes it easy to see that every combination of basis solutions containing a "red" basis solution cannot be strongly connected, due to the structure of each "red" basis solution with regard to CG. Consequently, we can focus on the "green" basis solutions (in this example, not in general) for building combinations which are strongly connected and therefore correspond to a non-decreasing closed path. However, not every combination of strongly connected basis solutions is strongly connected itself, e.g. `[[a6] c]` combined with `[[a9] c]` is not strongly connected. Such considerations can help to identify and analyze non-decreasing closed paths in a net, even beyond the particular context of stop-transitions.

3 Architecture and Installation

Cycl○n is implemented in Java and employs the data structures provided by the *APT*[9] library for the internal representation of Petri nets. Cycl○n also uses an algorithm of the APT library to determine a coverability graph of the net. The *GraphStream*[10] library is employed to visualize the internal data structures of the net and the coverability graph in an illustrative way. Cycl○n also uses an implementation of Tarjan's algorithm provided by the GraphStream library to determine the strongly connected components of a graph. Since the algorithm from [1] implemented in Cycl○n requires solving a homogeneous system of linear inequalities, the *polco tool*[11] library is employed to compute a basis of solutions for such a system of inequalities. While Cycl○n does not provide any editing capabilities for Petri nets, it allows to process PNML files created with tools like e.g. WoPeD[12] (which has also been employed for all our examples).

The Cycl○n tool is freely available for download at our website:
https://sttp1.fernuni-hagen.de/tools/cyclon/

The provided ZIP-archive file contains a runnable JAR file and several examples as PNML files (including all examples presented in this paper). Running Cycl○n merely requires an operation system with Java 8 (or higher) installed.

[8] considering a net of reasonable size, of course.
[9] APT – Analysis of Petri nets and labeled transition systems https://github.com/CvO-Theory/apt.
[10] GraphStream – A Dynamic Graph Library http://graphstream-project.org.
[11] polco Tool – Enumerate extreme rays of polyhedral cones https://csb.ethz.ch/tools/software/polco.html.
[12] WoPeD – Workflow Petri Net Designer http://www.woped.org.

4 Conclusion and Future Work

We presented the Cycl◌n tool and illustrated by several examples how it allows to determine stop-transitions of a Petri net in an easy way. We showed how the graphical representation and interactive features of Cycl◌n help to explain the results and make them comprehensible to the user.

In future work, we plan to extend the tool by editing capabilities for a Petri net under investigation, so that the effects of modifying certain elements of a net can be evaluated immediately without relying on external tools. Furthermore, we will work on replacing the current non-deterministic construction algorithm for a coverability graph by a more deterministic algorithm, so that every repeated coverability graph computation will result in the same graph structure (and the the same numbering of arcs).

References

1. Desel, J.: Can a single transition stop an entire net? In: International Workshop on Algorithms and Theories for the Analysis of Event Data 2019, ATAED@Petri Nets/ACSD 2019, Aachen, Germany, 25 June 2019, pp. 23–35 (2019). http://ceur-ws.org/Vol-2371/ATAED2019-23-35.pdf