PN2CCS: A Tool to Encode Petri Nets into Calculus of Communicating Systems

Benjamin Bogø[a,b,*], Andrea Burattin[a], Alceste Scalas[a]

[a] *DTU Compute, Technical University of Denmark, Denmark*
[b] *Department of Computer Science, University of Copenhagen, Denmark*

Abstract

PN2CCS is a software tool to encode Petri nets (PN) into the Calculus of Communication Systems (CCS). Its purpose is to allow Petri nets generated by most process mining algorithms (for instance, the α-miner) to be encoded into CCS, with the longer term goal of enabling the application tools and techniques developed for process calculi to the realm of process mining. PN2CCS is written in JavaScript and runs in modern web browsers with an interactive graphical user interface. The interface allows users to input a Petri net either by drawing it in the tool or importing a Petri net from a common file format for Petri nets. The tool then classifies the input Petri net and encodes it into CCS. The tool allows to encode a slight generalization of free-choice (workflow) nets as well as Petri nets directly expressible in CCS.

Keywords: Petri nets, Calculus of Communicating Systems, Encoding, Bisimulation, Free-choice workflow nets, Graphical user interface

*Corresponding author

Email addresses: bebo@di.ku.dk (Benjamin Bogø), andbur@dtu.dk (Andrea Burattin), alcsc@dtu.dk (Alceste Scalas)

Metadata

Nr.	Code metadata description	Please fill in this column
C1	Current code version	v1.2
C2	Permanent link to code/repository used for this code version	`https://github.com/bogoe/pn2ccs`
C3	Permanent link to Reproducible Capsule	`https://doi.org/10.5281/zenodo.15700968`
C4	Legal Code License	GNU General Public License (GPL), version 3+
C5	Code versioning system used	git
C6	Software code languages, tools, and services used	HTML, CSS and JavaScript
C7	Compilation requirements, operating environments and dependencies	Modern web browser (like Google Chrome and Mozilla Firefox)
C8	If available, link to developer documentation/manual	`https://github.com/Bogoe/pn2ccs/blob/main/README.md`
C9	Support email for questions	bebo@di.ku.dk

Table 1: Code metadata

1. Motivation and Significance

Petri nets and process calculi are among the most successful tools for the modelling and verification of concurrent systems. They have significantly different approaches and application fields: in particular, Petri nets are used to describe business workflow processes in *process mining*, whereas process calculi are used for static verification of behavioural properties (e.g. via type checking or (bi)simulations), often connected to programming languages. Most existing literature focus on encodings from process calculi to Petri nets while the other direction is less studied. [1]

In our theoretical paper [1], we present various encodings from different classes of place/transition Petri nets into the Calculus of Communication Systems [2], with focus on classes of Petri nets obtained via process mining algorithms. The purpose of such encodings is to establish a foundation that enables usage of analysis and verification techniques from the realm of process calculi in the realm of process mining. Having a tool that implements such encodings will help users to make experiments and give them a better understanding of the encodings. It will also be a stepping stone towards new applications for analysing mined processes with process calculi tools.

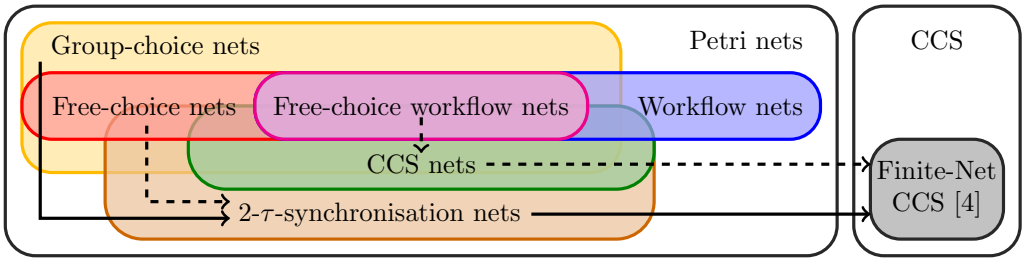

Figure 1: Modified version of Figure 1 in [1] that shows the relation between Petri net classes and the CCS fragment supported by PN2CCS. The arrows show which classes can be converted into which other classes. Only the solid arrows are supported by PN2CCS.

PN2CCS [3] is a tool that implements the encodings from Petri nets into CCS and closely follows the theoretical encoding framework developed in [1]. It supports its users through a graphical user interface (GUI). The initial version was published as companion artefact of the paper [1] for COORDINATION 2024. Figure 1 shows the Petri net classes that PN2CCS recognizes, and the arrows correspond to the encodings presented in [1]. At its core, PN2CCS implements the theory of [1] via (1) a transformation of group-choice nets into 2-τ-synchronisation nets (Algorithm 6 in [1]), and (2) an encoding from 2-τ-synchronisation nets into CCS (Algorithm 3 in [1]). This corresponds to the solid arrows in Figure 1, which are sufficient to encode all the encodable classes of Petri nets covered in [1].

Since its first public release, PN2CCS has been extended to also generate a viewable version of the intermediate Petri net (technically, a *2-τ-synchronisation net*) that PN2CCS generates during its encoding operation. It also features improvements to its GUI and better support for touch devices.

2. Software Description

PN2CCS is written in JavaScript and runs in modern web browsers with an interactive graphical user interface. Further details on how to run and use the tool is available in the file `README.md` in the artefact [3].

2.1. Software Architecture

PN2CCS is divided into three main components: the Petri net module, the CCS module and the graphical user interface (GUI). The Petri net module maintains a dynamic graph representation of Petri nets. It contains methods for updating, classifying and encoding Petri nets. The CCS module has an abstract syntax tree representation of the CCS and methods for converting it into a string. Finally, the GUI is responsible for the visual presentation of the Petri nets and for handling user inputs for manipulation of the Petri nets.

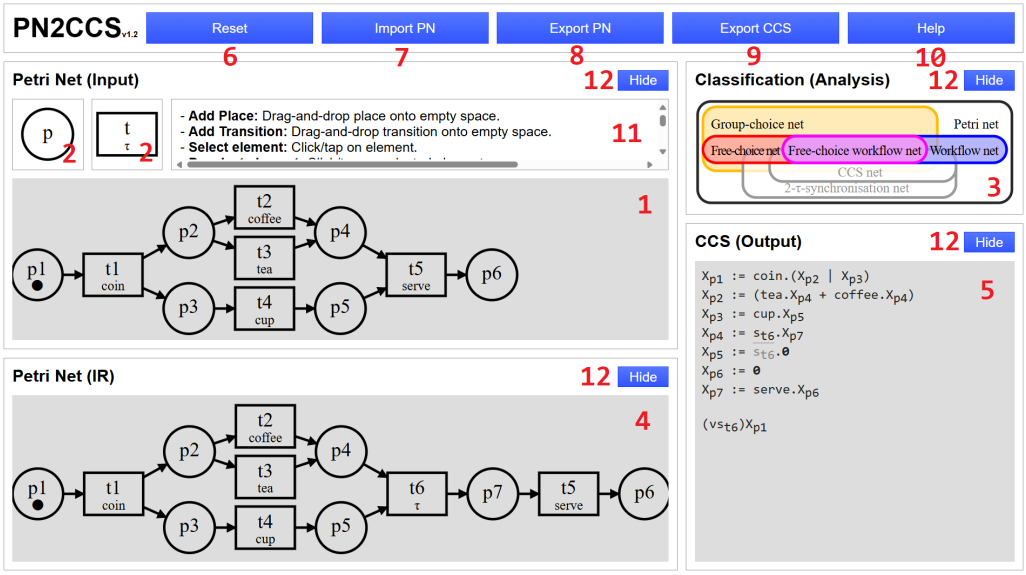

Figure 2: The graphical user interface in PN2CCS.

The GUI is responsive and the view for large screens is shown in Figure 2. For smaller screens, the elements are stacked and the screen becomes scrollable. The numbers in Figure 2 denote:

1. Area for drawing the input Petri net (the one that should be encoded).
2. Place/transition, draggable onto Area 1 when drawing a Petri net.
3. Classification of the Petri net in Area 1.
4. Area with the intermediate representation (IR) generated during the encoding of the Petri net in Area 1.
5. CCS obtained by encoding the Petri net in Area 1.
6. Button to reset the tool (clears the Petri net in Area 1).
7. Button to import Petri net from a PNML file [5].
8. Button to export the Petri net in Area 1 to a PNML file.
9. Button to export the CCS in Area 5 as a text file.
10. Button to get detailed help about the tool.
11. List that briefly describes how to draw/edit the Petri net in Area 1.
12. Buttons that can be used to show/hide parts of the tool.

2.2. Software Functionalities

PN2CCS takes a Petri net as input and produces a CCS process as output. The input Petri net can either be imported from a PNML file by using the import functionality (Button 7 in Figure 2) or drawn manually in the

GUI. Petri nets can be drawn in PN2CCS by drag-and-dropping places and transitions from Area 2 in Figure 2 onto the drawing Area 1. Edges can be added by left-clicking on a place and then on a transition, or *vice versa*. Places and transitions can be moved by selecting them (left-click) and then dragging them to their new location (left-click again to deselect). The number of tokens in a place or the action of a transition can be changed by right-clicking on the place or transition. Places, transitions and edges can be deleted by right-clicking on them and then select the delete button in the dialog. Please consult the help screen in the tool (Button 10 in Figure 2) for more details and instructions for touch devices.

Whenever the input Petri net is updated, it is classified in terms of the classes in Figure 1. The matching classes are coloured in Area 3 in Figure 2. The definitions of the classes and how they are detected are described below:

- **CCS net** (Definition 12 in [1]): All transitions must have 1 or 2 ingoing edges and every transition with 2 ingoing edges must have label τ.
- **2-τ-synchronisation net** (Definition 13 in [1]): Same as CCS net, except that transitions with zero ingoing edges are also allowed.
- **Free-choice net** (Definition 10 in [1]): All places with multiple outgoing edges must only have edges to transitions with one ingoing edge.
- **Workflow net** (Definition 9 in [1]): There must be exactly one source place i with no ingoing edges and exactly one sink place o with no outgoing edges. In addition, every place/transition must be on a directed path from i to o. PN2CCS checks this by running a breadth-first search (BFS) in forward direction from i and backwards from o, and then check that all nodes were reached in both runs.
- **Free-choice workflow net** (Definition 11 in [1]): Both checks for free-choice net and workflow net must pass.
- **Group-choice net** (Definition 15 in [1]): All places with an edge to a same transition, all have edges to the exact same transitions.

If the input Petri net is classified as a group-choice net but not a 2-τ-synchronisation net, it is transformed into a 2-τ-synchronisation net (the IR), and shown in Area 4 on Figure 2. Internally, PN2CCS performs this transformation as follows: (1) select a transition t that violates the 2-τ-synchronisation net constraints; (2) remove all outgoing edges from all places with an edge to t; (3) synchronise these places two places at a time until only one place remains; (4) and connect the last place to all transitions that had edges removed. The pairwise-synchronisation is achieved by connecting two places at a time to a new transition t' with label τ leading to a new place p'.

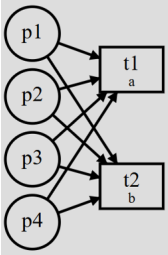

Figure 3: Group-choice net.

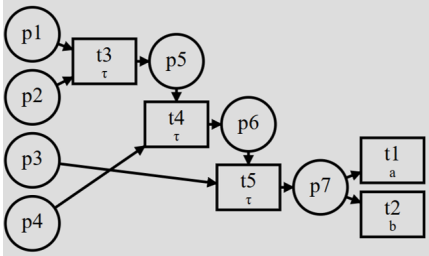

Figure 4: Sequential synchronisation pattern of Figure 3 that synchronises one original place at a time.

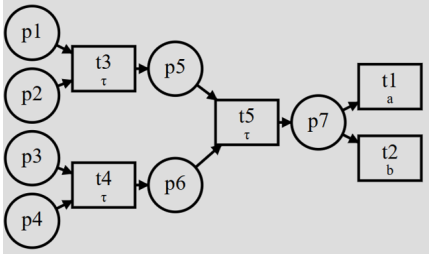

Figure 5: Parallel synchronisation pattern of Figure 3 that synchronises the original places in pairs.

For instance, when Figure 3 (`examples/pnml/s03-sync-patterns.pnml` in the artefact [3]) is imported into PN2CCS, it may randomly produce either sequential synchronisation patterns as in Figure 4, or more parallel patterns like in Figure 5 when transforming it. This transformation is trivial in the Petri net model but it is more involved for the GUI, as it also has to assign a position to the new places and transitions. PN2CCS handles this by pushing selected places and transitions to the right.

When the input Petri net has been transformed into a 2-τ-synchronisation net, the latter is encoded into CCS as described for the example in Section 3.

3. Illustrative Examples

The artefact [3] contains several Petri nets (directory `examples/pnml`) that one can import and modify in PN2CCS – including examples covering all the classes in Figure 1, and some Petri nets from process mining literature. Details are provided in the file `README.md`.

In this section we use the *order-to-cash* Petri net in Figure 6 (based on Fig. 3.6 in [6]) to show how PN2CCS works. The Petri net describes how an order for a product is processed. It starts by checking if the products are in stock (t_1) and if not, the order is rejected (t_9). Otherwise, the product

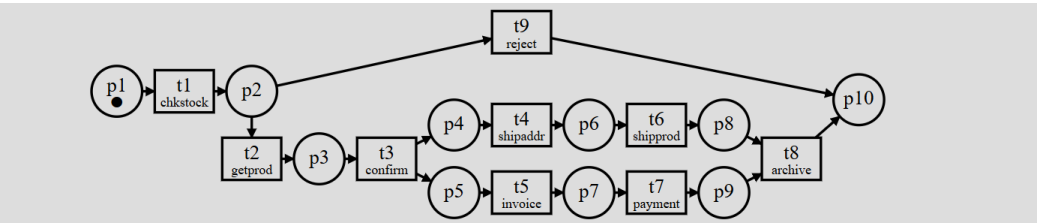

Figure 6: Free-choice workflow net that describes an *order-to-cash process*.

is fetched from the warehouse (t_2) and the order is confirmed (t_3). Afterwards, the shipping address is fetched (t_4) and the product is shipped to that address (t_6). Meanwhile, an invoice is sent (t_5) and the payment is received (t_7). When both the product is shipped and the payment is received, then the order is archived (t_8).

The Petri net in Figure 6 is available as a PNML file in the artefact [3] (`examples/pnml/s06-order-to-cash-process.pnml`) and can be imported in PN2CCS by using the import functionality (Button 7 in Figure 2). The Petri net can also be drawn manually by following the step-by-step instructions in the file `README.md` [3].

When the complete *order-to-cash* Petri net has been drawn or imported, PN2CCS classifies it as a free-choice workflow net (and therefore, also a group-choice net, by Figure 1), but not a 2-τ-synchronisation net (and therefore, also not a CCS-net, by Figure 1). In fact, Figure 6 is not a 2-τ-synchronisation net because t_8 has two ingoing edges but its label is not τ. All places in Figure 6 pass the group-choice net check: in fact, they only have edges to transitions with one ingoing edge — except p_8 and p_9 that *both* only have an edge to t_8.

To continue the encoding, PN2CCS transforms the Petri net in Figure 6 into a (weakly bisimilar [1]) 2-τ-synchronisation net by removing the outgoing edges from p_8 and p_9, and connecting them to a new τ-transition t_{10} that is connected to the original transition t_8 via a new place p_{11}. The result is shown in Figure 7.

Finally, the Petri net in Figure 7 is encoded into the CCS process in Figure 8. Every place is encoded as a choice between its transitions, for instance p_2 is encoded into the process X_{p_2} with the choice between t_2 with action `getprod` followed by a token to p_3 (X_{p_3}) and t_9 with action `reject` followed by a token to p_{10} ($X_{p_{10}}$). The place p_3 only has one option, namely t_3, which produces tokens to both p_4 and p_5: this is encoded as the parallel composition ($X_{p_4} \mid X_{p_5}$). Place p_{10} has no options, so the choice collapses to inaction ($\mathbf{0}$). Transitions with two ingoing edges require a τ-synchronisation,

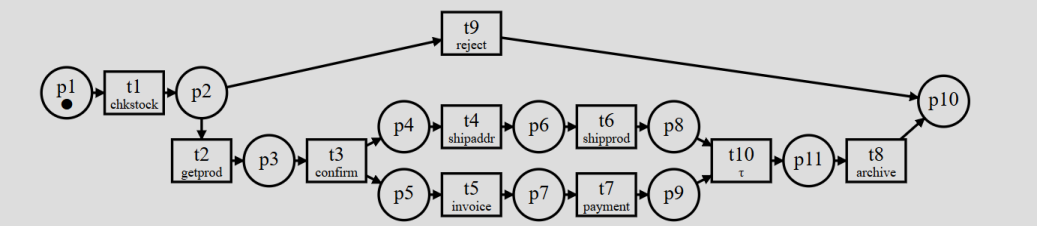

Figure 7: The 2-τ-synchronisation net (IR) obtained by transforming Figure 6.

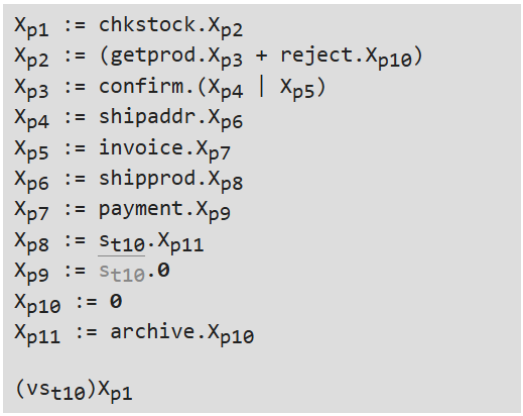

Figure 8: CCS obtained by encoding the 2-τ-synchronisation net in Figure 7.

so p_8 and p_9 with an edge to t_{10} has a new action s_{t10} and its co-action $\overline{s_{t10}}$; the new action is restricted (νs_{t10}) in the initial process at the bottom. The initial process is the parallel composition of each place repeated once per token: in Figure 7, only p_1 has a token so it simply becomes X_{p_1}. If more tokens are added to the place p_1, then the process X_{p_1} is replicated accordingly, for instance 3 tokens yields $X_{p_1}^3$.

4. Impact

PN2CCS is part of a larger line of research which aims at studying the application of techniques from the realm of process calculi (e.g. behavioural equivalences and preorders, model checking) to the analysis and verification of Petri nets obtained via process mining. PN2CCS implements the first stepping stone for this line of research, i.e., the translation of mined Petri nets into behaviourally-equivalent CCS processes (using the encodings introduced in [1]). Crucially, PN2CCS supports PNML files generated by standard process mining tools in literature, such as the α-miner [7] — and several such example nets are included in the artefact.

While many behavioural properties such as deadlock-freeness or reachability can also be analyzed directly on Petri nets, encoding the result of process mining into CCS enables the reuse of a rich ecosystem of tools, specifically developed for process calculi. For example, after encoding the order-to-cash process from Figure 6, an organization may verify its weak bisimilarity against a reference process that they are planning to implement. Tools like the mCRL2 model checker [8] can also allow querying the process for properties that are not naturally supported in Petri nets e.g., "on all paths, a payment is eventually received before archiving the order".

5. Future Plans

We plan to use PN2CCS with mined nets from larger case studies and explore how analysis techniques from the realm of process calculi can be applied and adapted for process mining purposes. To this end, it could be useful to let the user choose the synchronisation pattern when generating the 2-τ-synchronisation net. PN2CCS could also be extended with export options to other tools: e.g., it could be easily adapted to produce its output in the process calculus of the mCRL2 model checker [8].

References

[1] B. Bogø, A. Burattin, A. Scalas, Encoding petri nets into ccs, in: I. Castellani, F. Tiezzi (Eds.), Coordination Models and Languages, Springer Nature Switzerland, Cham, 2024, pp. 38–55. doi:10.1007/978-3-031-62697-5_3.
URL https://doi.org/10.1007/978-3-031-62697-5_3

[2] R. Milner, A Calculus of Communicating Systems, Vol. 92 of Lecture Notes in Computer Science, Springer, 1980. doi:10.1007/3-540-10235-3.
URL https://doi.org/10.1007/3-540-10235-3

[3] B. Bogø, Encoding petri nets into ccs (Dec. 2024). doi:10.5281/zenodo.15700968.
URL https://dtu.bogoe.eu/tools/pn2ccs/

[4] R. Gorrieri, C. Versari, Introduction to Concurrency Theory - Transition Systems and CCS, Texts in Theoretical Computer Science. An EATCS Series, Springer, 2015. doi:10.1007/978-3-319-21491-7.
URL https://doi.org/10.1007/978-3-319-21491-7

[5] J. B. et al., Pnml grammar, version 2009 (2009).
URL https://www.pnml.org/version-2009/version-2009.php

[6] M. Dumas, M. La Rosa, J. Mendling, H. A. Reijers, Essential Process Modeling, Springer Berlin Heidelberg, Berlin, Heidelberg, 2018, pp. 75–115. doi:10.1007/978-3-662-56509-4_3.
URL https://doi.org/10.1007/978-3-662-56509-4_3

[7] W. M. P. van der Aalst, T. Weijters, L. Maruster, Workflow mining: Discovering process models from event logs, IEEE Trans. Knowl. Data Eng. 16 (9) (2004) 1128–1142. doi:10.1109/TKDE.2004.47.

[8] O. Bunte, J. F. Groote, J. J. A. Keiren, M. Laveaux, T. Neele, E. P. de Vink, W. Wesselink, A. Wijs, T. A. C. Willemse, The mcrl2 toolset for analysing concurrent systems - improvements in expressivity and usability, in: T. Vojnar, L. Zhang (Eds.), Tools and Algorithms for the Construction and Analysis of Systems - 25th International Conference, TACAS 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings, Part II, Vol. 11428 of Lecture Notes in Computer Science, Springer, 2019, pp. 21–39. doi:10.1007/978-3-030-17465-1_2.
URL https://doi.org/10.1007/978-3-030-17465-1_2