Synthetic Portnet Generation with Controllable Complexity for Testing and Benchmarking

Madiou Diallo, Benny Akesson, Debjyoti Bera, and Ronald Begeer

Problem

Petri nets are a nice basis to model and reason about the behavior of asynchronous communication systems

- Various tooling exists for modelling and analysis of different classes of nets

Software interfaces can be efficiently modelled using the class of portnets

- A constrained type of open nets that provide guarantees on weak termination

Generation of synthetic models is helpful when testing or benchmarking analysis/synthesis tools

- Large sets of random models with user-specified characteristics
- Tools for generating such models exist, but not for portnets

Overview of Contributions

Paper has four contributions related to synthetic portnet generation

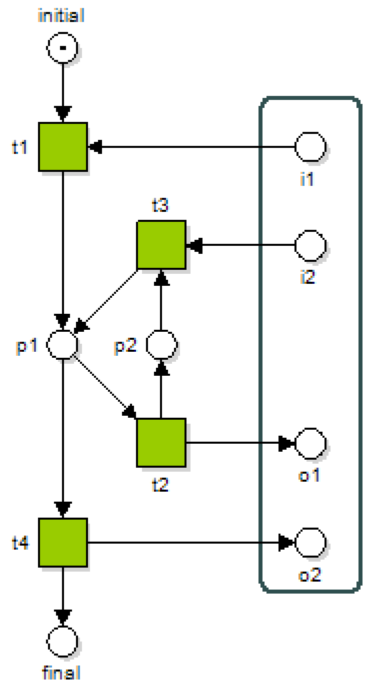
1. **Defined the notion of complexity within portnets using three parameters**
    - Inputs, outputs, and prevalence of non-determinism
2. **A method for synthetic generation of portnets using the complexity metrics as input**
3. **An implementation as an open-source Python tool**
4. **Experimental evaluation of method and demonstration of relation between the input of complexity parameters and the resulting portnet**

Preliminaries

Portnets

Constrained state machine open workflow nets suitable for modelling interfaces

- Communication protocols are **state machines**
- An interaction between a server and a client is a **workflow** with a single initial and final place, where all nodes are on a path between those places
- Open nets contain a set of **interface places** (inputs and outputs)



Key constraints

- Each transition is connected to **exactly one** interface place, and vice versa
- **Leg Property**: Each leg must have at least one send and one receive transition
- **Choice Property:** All transitions in the postset must communicate in same direction

A portnet composed with a mirrored client is guaranteed to be weakly terminating

Refinement Rules

Four refinement rules utilized for portnet refinement

- Each rule consists of a **base rule** and **modified rules**

Base rule form the starting point for refinement

- Refines a place or a transition

Modified rules ensure portnet constraints adhered to after application of a base rule

- E.g. determine the direction of communication

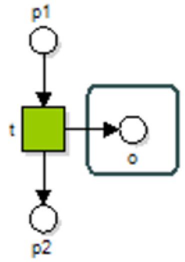
Rules are denoted as Rx modified rules as Rx'/Rx"

Four Refinement Rules: Deterministic

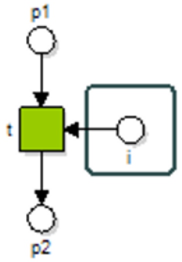
Start

$R0$

$R0'$

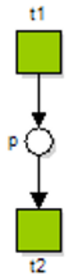


$R0''$

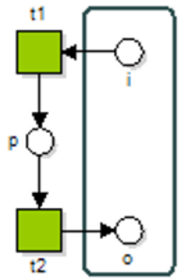


Start

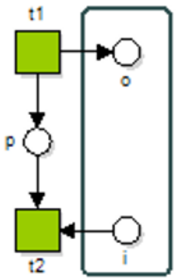
$R1$



$R1'$

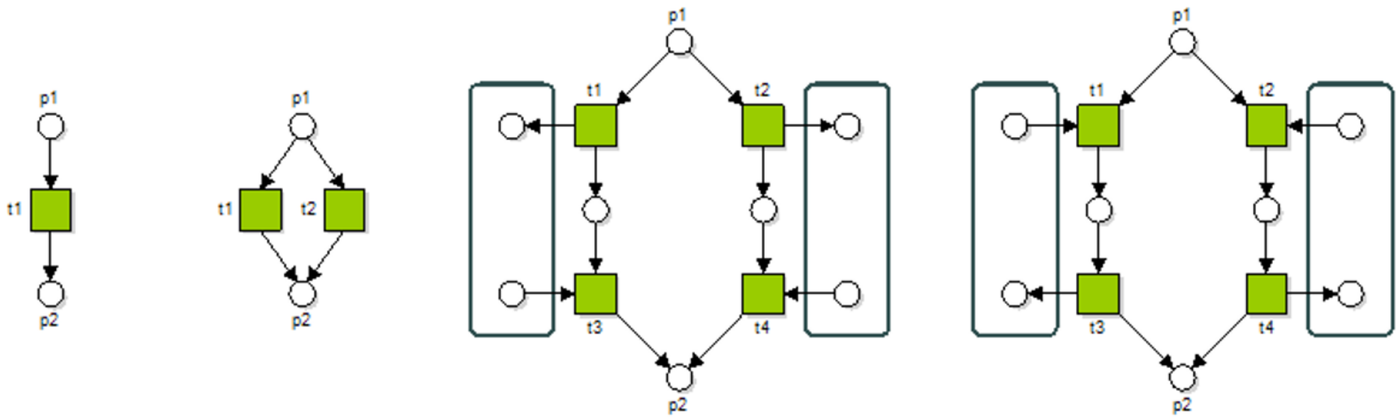


$R1''$

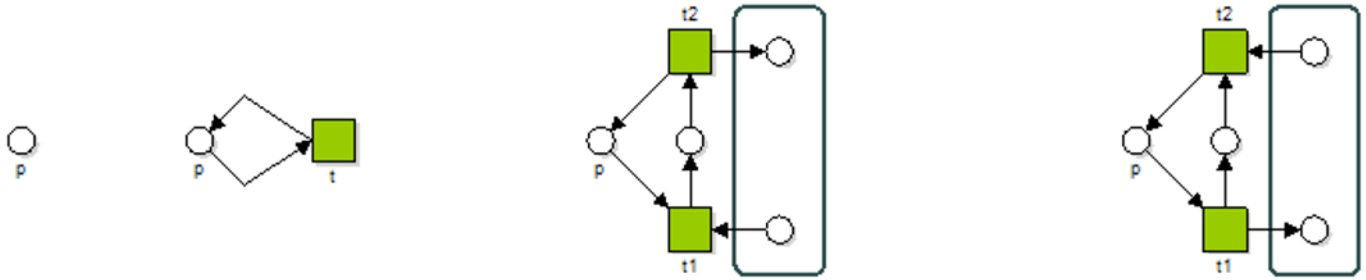


Four Refinement Rules: Non-Deterministic

Start *R2* *R2'* *R2''*



Start *R3* *R3'* *R3''*



Complexity Parameters

Complexity Parameters

Complexity parameters allow one to argue about the structure of a given portnet

- Used to control the output of generation

Notion of complexity in three parameters

- Number of inputs and outputs, and the prevalence of non-determinism

Prevalence is defined as the fraction of arcs originating from split places

$$\rho(N) = \frac{|\{(p,t) \in F \mid p \in P \wedge \mid p^{\bullet} \mid > 1\}|}{|\{(p,t) \in F \mid p \in P\}|}$$

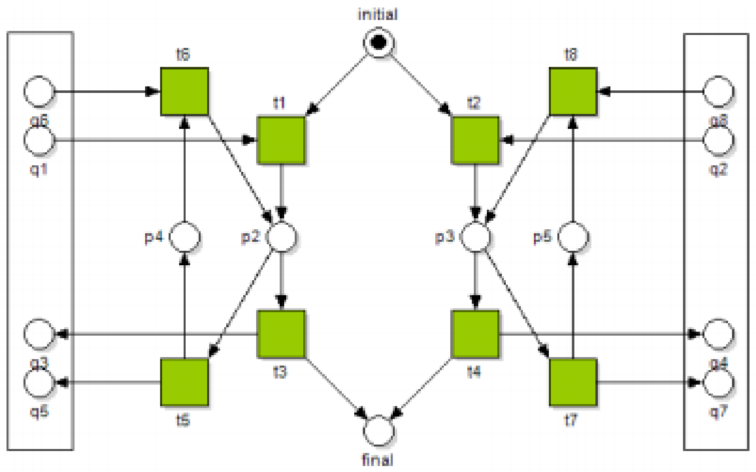


Fig. 3: Example portnet with characteristics ($I_{exp} = 4, O_{exp} = 4, Pr_{exp} = 0.75$)

Portnet Generation Method

Portnet Generation Method

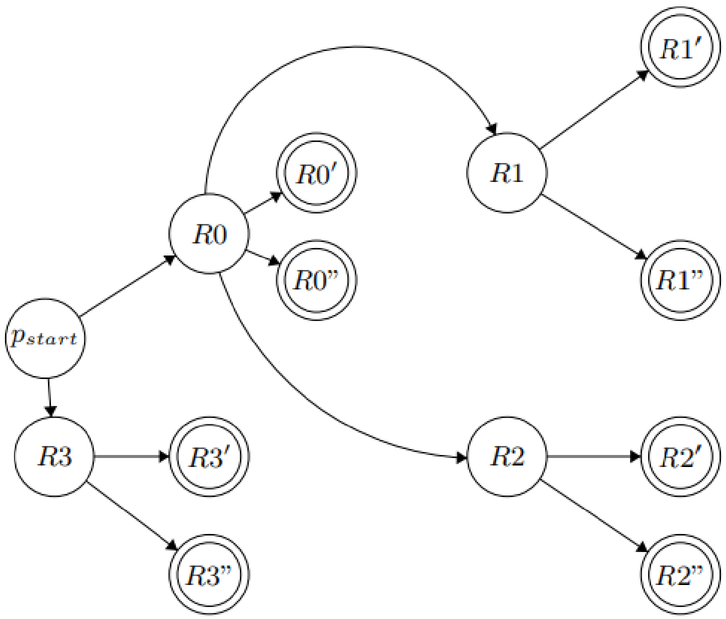
The Allowed Ruleset determines how refinement rules can be applied in sequence

- A sequence of refinements is referred to as a **refinement iteration**

Generation algorithm uses allowed ruleset and provided complexity parameters to generate resulting structure

- Each rule affects parameter values differently

Each refinement iteration starts from a single initial place and ends with a modified rule



Synthetic Portnet Generation Algorithm

The generation algorithm results in a randomized portnet controlled by the complexity parameters

1. **Start from the Initial place (L1)**
2. **If inputs, outputs not reached (L8)**
3. **Randomized refinement iteration (L10-14)**
   - From non-deterministic or deterministic ruleset
     (dependent on the current prevalence of the net under generation)
4. **Subtract inputs, outputs from the current (L13-14)**
5. **Back to step 2, if necessary**

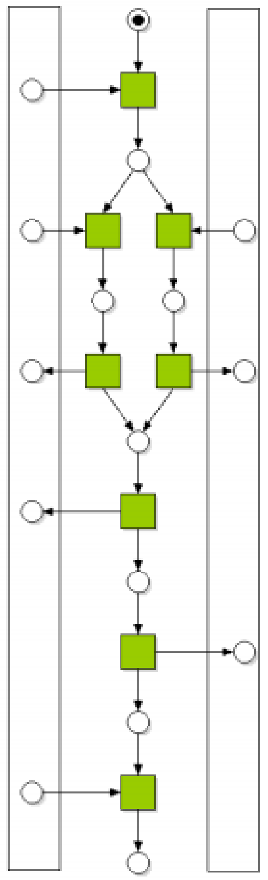
Algorithm 1 Synthetic Portnet Generation

Inputs: Expected number of inputs I_{exp}, outputs O_{exp}, and prevalence Pr_{exp}

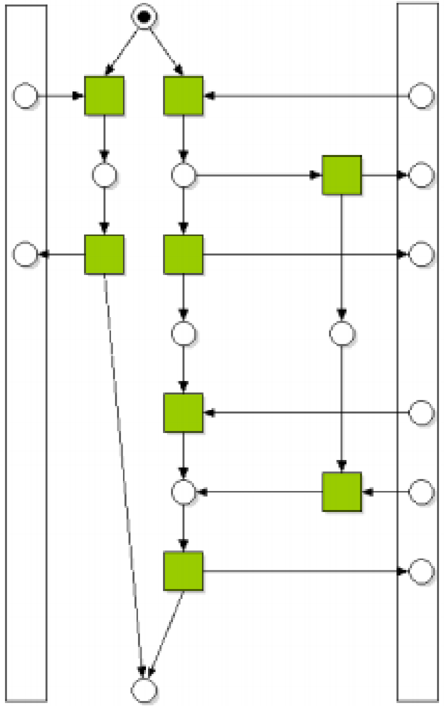
Output: *Portnet* N approximating the inputs.

1: Let portnet N with exactly one place and this is the *initial* place.
2: Let $I_{cur} \leftarrow I_{exp}$, $O_{cur} \leftarrow I_{exp}$, $Pr_{cur} \leftarrow 0$
3: Let $detrules = \{\langle R0, R0' \rangle, \langle R0, R0" \rangle, \langle R0, R1, R1' \rangle, \langle R0, R1, R1" \rangle\}$
4: ▷ set of sequences of deterministic refinement rules
5: Let $nondetRules = \{\langle R0, R2, R2' \rangle, \langle R0, R2, R2" \rangle, \langle R3, R3' \rangle, \langle R3, R3" \rangle\}$
6: ▷ set of sequences of non-deterministic refinement rules
7: Let *ruleset* be an empty *sequence* of refinement rules
8: **while** I_{cur} and O_{cur} are not equal to zero **do**
9: $Pr_{cur} \leftarrow \rho(N)$
10: **if** $Pr_{cur} < Pr_{exp}$ and $I_{cur} \geq 1$ and $O_{cur} \geq 1$ **then** $ruleset \leftarrow nondetrules$
11: **else** $ruleset \leftarrow detrules$
12: Pick $r \in ruleset$ and $p \in P_N$, such that (p is not *initial* or *final* and $r(0) \neq R3$) and (I_{cur} and O_{cur} are greater than or equal to the inputs and outputs added to the net introduced by r)
13: Subtract the number of inputs introduced by rule r from I_{cur}
14: Subtract the number of outputs introduced by rule r from O_{cur}
15: **while** r is not empty sequence **do**
16: **if** $r(0) = R3'$ or $r(0) = R3''$ **then**
17: **if** refining place p with $R3'$ causes a choice property violation **then**
18: $r(0) \leftarrow R3''$
19: **else** $r(0) \leftarrow R3'$
20: Refine place p of portnet N with rule $r(0)$.
21: Remove rule $r(0)$

Example Generations



(a) Generated portnet with parameters ($I_{exp} = 4, O_{exp} = 4, Pr_{exp} = 0.3|Pr = 0.25$)



(b) Generated portnet with ($I_{exp} = 4, O_{exp} = 4, Pr_{exp} = 0.5|Pr = 0.5$)

Experiments

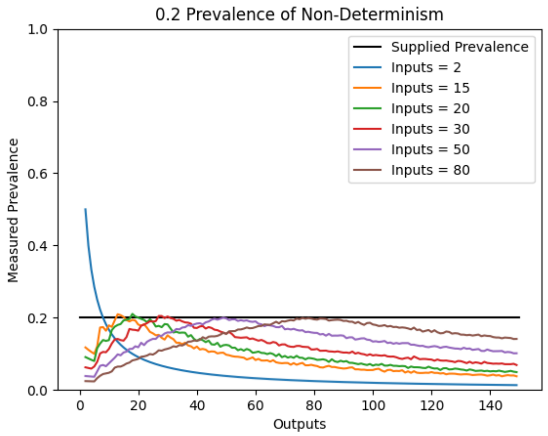
Experiments

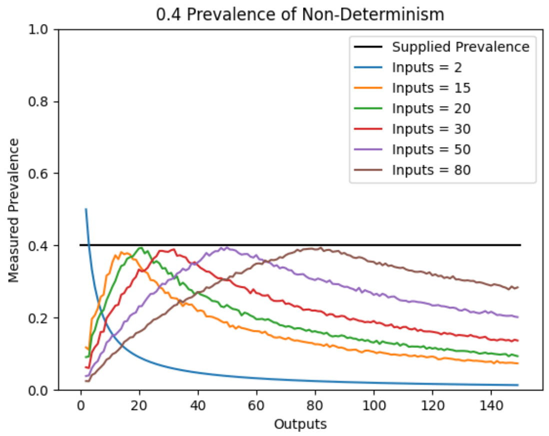
Experimentally display the inherent relation between user-specified complexity parameters and the extent to which the generator can satisfy them

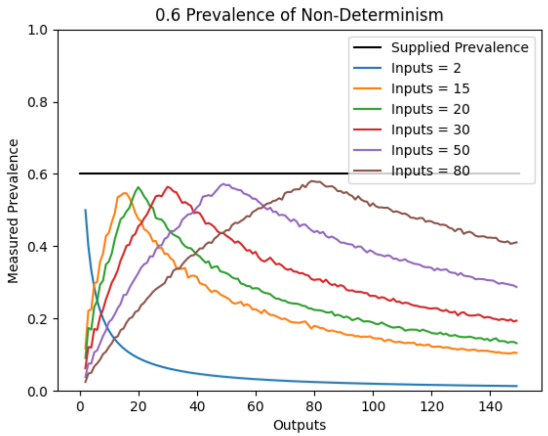
Four conducted experiments with fixed prevalence of non-determinism {0.2, 0.4, 0.6, 0.8}

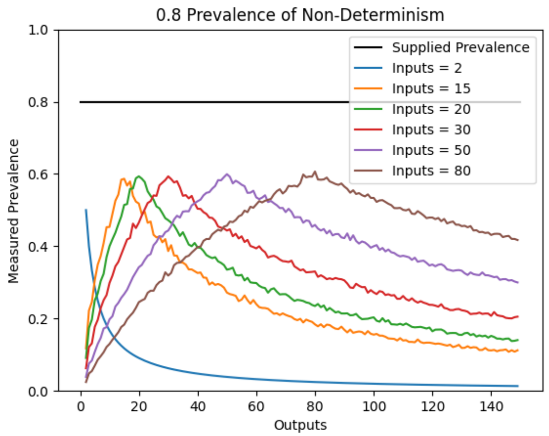
- Varying inputs and outputs {2, 15, 20, 30, 50, 80}
- Measure the average observed prevalence as a function of user-supplied complexity parameters
- 40 iterations

Findings from Experiments









Result of Experiments

Experiments result in some key observations regarding generation

Difference between inputs and outputs result in a decreasing average prevalence

- Inherent relation between the refinement rules and complexity parameters

Average observed prevalence does not exceed ~0.6

- Result of randomly selecting refinement rules & application onto place

Conclusions

Conclusions

Introduced a methodology for synthetic generation of portnets of various complexity

- Benefit to those requiring input for tooling within the context of modelling and analysis
- Allows for easier testing and benchmarking using portnets as input

Introduced a definition of portnet complexity

- Number of inputs and outputs, and prevalence of non-determinism

Experiments showing the relation between refinement rules and complexity parameters

Methodology implemented as an open-source Python tool

- Output as PNML representation