Synthetic Portnet Generation with Controllable Complexity for Testing and Benchmarking

Madiou Diallo, Benny Akesson, Debjayoti 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 \( R_x' / R_x'' \)
Four Refinement Rules: Deterministic
Four Refinement Rules: Non-Deterministic
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

\[
\rho(N) = \frac{\left| \left\{ (p,t) \in F \mid p \in P \land |p^*| > 1 \right\} \right|}{\left| \left\{ (p,t) \in F \mid p \in P \right\} \right|}
\]

Fig. 3. Example portnet with characteristics \((I_{exn} = 4, O_{exn} = 4, F_{exn} = 0.75)\)
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
Example Generations

(a) Generated portnet with parameters ($I_{inp} = 4, O_{inp} = 4, Pr_{inp} = 0.3/P_r = 0.25$)

(b) Generated portnet with ($I_{inp} = 4, O_{inp} = 4, Pr_{inp} = 0.5/P_r = 0.5$)
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