Formal Verification of Systems with Unbounded Agents

Tephilla Prince, Indian Institute of Technology Dharwad

The complexity and size of software has increased with time. And with it, the necessity to provide mathematical guarantees to prove the behaviour of the software with respect to its requirements has increased. Empirical studies show that the sooner the errors are identified and rectified in the software development lifecycle, the lesser the error costs. If there are errors in the design of the software, they propagate to errors in the implementation which become tedious to resolve. Being able to verify the specifications of these systems in the design stage of the software development (such as using bounded model checking [1]) is an important activity that can be done using formal techniques. The characteristics of the software play a role in the choice of the formal technique to verify a software. Client server systems are one of the largest programming paradigms. Crypto exchanges with an unbounded number of investors, multiplayer games where the number of players are not known apriori and services with an unbounded customer base are instances of unbounded client server systems.

We focus on client server systems with single server and unboundedly many clients, where the number of clients are not known apriori and there can be unbounded concurrent interactions between the clients and server. The major challenges are to identify the suitable model to abstract the behaviour of the systems, to identify suitable logics to specify the properties and to identify formal techniques for verifying the properties on the model. First, we model unbounded client-server systems as unbounded Petri nets [2,3] and their temporal properties are expressed in Linear Temporal Logic (LTL) with integer arithmetic. In this system, there is unboundedness in two dimensions, namely, the number of clients as well as in the client server interactions. Hence, we propose two dimensional bounded model checking (2D-BMC) for exploiting the unboundeness in these two dimensions and utilize SMT solvers for verification.

Second, when the unbounded client-server systems have distinguishable clients, unbounded Petri nets are no longer suitable to model them. Hence, we explore suitable concurrent models where the tokens are distinguished such as ν-nets [4] as shown in Fig. 1.

Definition 0.1. *Given an arbitrary set A, we denote by $\mathcal{MS}(A)$, the set of finite multisets of A, given by the set of mappings $m : A \to \mathbb{N}$. We denote by $S(m)$ the support of m, defined as follows: $S(m) = \{a \in A \mid m(a) > 0\}$. Distinguishable tokens (identifiers) are taken from an arbitrary infinite set* Id*. To handle this, we add matching variables labeling the arcs, taken from a set Var. To handle the movement of tokens inside the ν-net, we employ a finite set of variables, Var using which we label the arcs of the net. A ν-net is a coloured Petri net $N = (P, T, F)$, where*

- *P and T are finite disjoint sets of places and transitions, respectively,*
- *F: $(P \times T) \cup (T \times P) \to \mathcal{MS}(Var)$ defines the set of arcs of the net, satisfying $\nu \notin pre(t)$ for every $t \in T$.*

For a transition t of the net, we define, $post(t) = \bigcup_{p \in P} S(F(t,p))$, $pre(t) = \bigcup_{p \in P} S(F(p,t))$ and $Var(t) = pre(t) \bigcup post(t)$.

For instance, in Fig. 1, $T = \{t_{acc}, t_{rej}, t_{s\_exit}, t_{u\_exit}, t_{acc\_sink}, t_{rej\_sink}\}$ and $P = \{ p_{PR}, p_{SR}, p_{OP}, p_{PU}, p_{ES}, p_{EU}\}$. $pre(t_{acc}) = \{s, c\}$ and $post(t_{acc}) = \{s, c\}$, hence $Var(t_{acc}) = \{s, c\}$.

We also require a suitable logic, which is richer than LTL, to represent their properties naturally, such as a variant of First Order logic. The 2D-BMC algorithm is used to automatically verify the system with these extensions and implemented as a prototype.

Third, clients have their own internal structure. We model each client as a Petri net which interacts with the server Petri net. The interaction of unbounded client server systems can be represented as client Petri nets nested within the server Petri net as in Fig. 2. This is a type of higher order Petri nets, called Elementary Object Systems (EOS) [5,6].

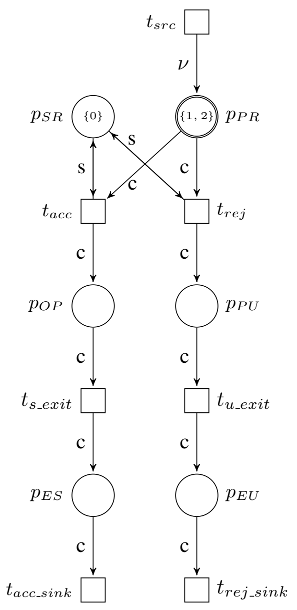

Figure 1: A restricted ν-net modeling APS

Definition 0.2 (EOS). *An* EOS $\mathfrak{E}$ *is a tuple* $\mathfrak{E} = \langle \hat{N}, \mathcal{N}, d, \Theta \rangle$ *where:*

1. $\hat{N} = \langle \hat{P}, \hat{T}, \hat{F} \rangle$ *is a PN called* system net*;* $\hat{T}$ *contains a special set* $ID_{\hat{P}} = \{id_p \mid p \in \hat{P}\} \subseteq \hat{T}$ *of* idle transitions *such that, for each distinct* $p, q \in \hat{P}$*, we have* $\hat{F}(p, id_p) = \hat{F}(id_p, p) = 1$ *and* $\hat{F}(q, id_p) = \hat{F}(id_p, q) = 0$.
2. $\mathcal{N}$ *is a finite set of PNs, called* object PNs*, such that* $\blacksquare \in \mathcal{N}$ *and if* $(P_1, T_1, F_1), (P_2, T_2, F_2) \in \mathcal{N} \cup \hat{N}$*, then* $P_1 \cap P_2 = \emptyset$ *and* $T_1 \cap T_2 = \emptyset$.
3. $d : \hat{P} \to \mathcal{N}$ *is called the* typing function.
4. Θ *is a finite* set of events *where each* event *is a pair* $(\hat{\tau}, \theta)$*, where* $\hat{\tau} \in \hat{T}$ *and* $\theta : \mathcal{N} \to \bigcup_{(P,T,F) \in \mathcal{N}} T^{\oplus}$*, such that* $\theta((P, T, F)) \in T^{\oplus}$ *for each* $(P, T, F) \in \mathcal{N}$ *and, if* $\hat{\tau} = id_p$*, then* $\theta(d(p)) \neq \emptyset$.

We also explore the possibility of losing information in the EOS at various nesting levels and their effects on the reachability and coverability problems and chart their decidability status. We built a tool to verify properties such as safety, deadlock and reachability properties using LTL on the standard PN and EOS, as well as their lossy counterparts.

This gives us a suite of formal verification tools for client server systems with unbounded clients using various concurrent models and various suitable logics to represent their properties.

References

[1] Armin Biere, Alessandro Cimatti, Edmund M. Clarke, Ofer Strichman, and Yunshan Zhu. Bounded model checking. *Adv. Comput.*, pages 117–148, 2003.

[2] T. Murata. Petri nets: Properties, analysis and applications. *IEEE*, 77(4):541–580, 1989.

[3] Carl Adam Petri. Kommunikation mit Automaten. Dissertation, Schriften des IIM 2, Rheinisch-Westfälisches Institut für Instrumentelle Mathematik an der Universität Bonn, Bonn, 1962.

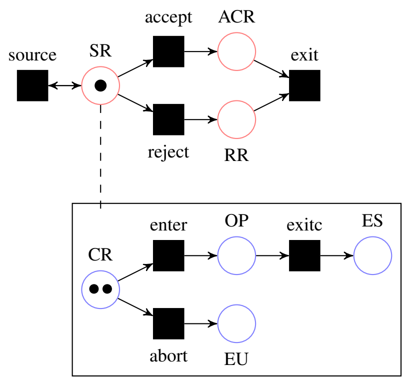

Figure 2: EOS modeling the single server (unbounded) multiple client system

[4] Fernando Rosa-Velardo and David de Frutos-Escrig. Name creation vs. replication in Petri net systems. *Fundam. Informaticae*, 88(3):329–356, 2008.

[5] Rüdiger Valk. Nets in computer organization. In *Petri Nets: Applications and Relationships to Other Models of Concurrency*, pages 218–233, Berlin, Heidelberg, 1987. Springer Berlin Heidelberg.

[6] Rüdiger Valk. Object Petri nets: Using the nets-within-nets paradigm. In *Lectures on Concurrency and Petri Nets, Advances in Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*, pages 819–848. Springer, 2003.