Improving Saturation Efficiency with Implicit Relations[†]

Shruti Biswal   Andrew S Miner

Iowa State University

sbiswal@iastate.edu
asminer@iastate.edu

Petri Nets 2019

[†]Supported in part by the National Science Foundation (ACI-1642397)

State-space Generation

- High-level formalisms model real world discrete-state systems
- Formal verification of systems may require exhaustive analysis of entire *reachability set*, which can be generated using :
  - Explicit techniques : explore one state at a time
  - Symbolic methods, like *saturation* : explore sets of states
- Fast reachability set generation $\rightsquigarrow$ Accelerated system analysis
- Traditional *saturation* implementation includes :
  - Set of reachable states encoded using *MDDs*
  - State transitions encoded using *2L-MDDs* or *MxDs*

Objective

- Underlying data structures in *saturation* affect its efficiency
- *Implicit relation forests* : Alternative for encoding transitions
  - Applicable to a sub-class of high-level discrete-state models
  - Static representation, one-time construction
  - Memory and computation efficient w.r.t MxDs

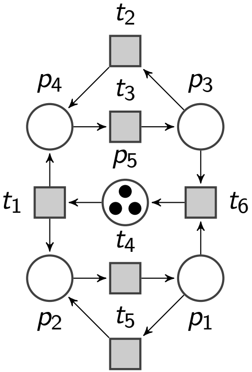
Class of Models

- Domain of finite discrete-state model $\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathbf{i}_0, \Delta)$
  - $\mathcal{V} = \{v_1, v_2, \ldots, v_L\}$ is a set of *state variables* of the model.
  - $\mathcal{E} = \{e_1, e_2, \ldots, e_{|\mathcal{E}|}\}$ is a finite set of *events* of the model.
  - $\mathbf{i}_0 \in \mathbb{N}^L$ is the initial state of the model.
  - $\Delta : \mathbb{N}^L \times \mathcal{E} \nrightarrow \mathbb{N}^L$ is the next state (partial) function such that

$$\Delta((i_1, \ldots, i_L), e) \;=\; \mathcal{N}_e(\mathbf{i}_0) \;=\; (\Delta_{e,1}(i_1), \ldots, \Delta_{e,L}(i_L))$$

    where for any $k \in [1, L]$, $\Delta_{e,k}$ is a *local* next state function and $\Delta_{e,k}(i_k) \geq 0$.
- Existing formalisms $\in \mathcal{M}$:
  - Ordinary PNs, with inhibitor and reset arcs
  - PN with marking-dependent arc cardinalities and/or transition guards : Must have *deterministic Kronecker-consistent* events

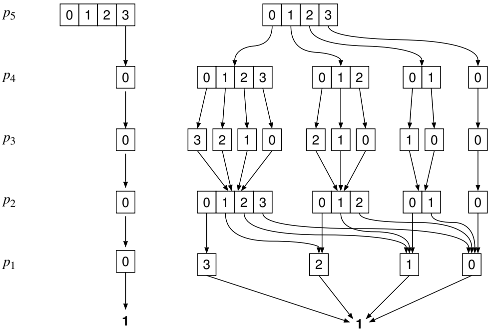
Example : Fork-Join Petri Net



- $\mathcal{V} = \{p_1, p_2, p_3, p_4, p_5\}$
- $\mathcal{E} = \{t_1, t_2, t_3, t_4, t_5, t_6\}$
- $\mathbf{i}_0 = (0, 0, 0, 0, 3)$
- Δ :
  - $\Delta\big((i_1, i_2, i_3, i_4, i_5 > 0), t_1\big) = (i_1, i_2 + 1, i_3, i_4 + 1, i_5 - 1)$
  - $\Delta\big((i_1, i_2, i_3 > 0, i_4, i_5), t_2\big) = (i_1, i_2, i_3 - 1, i_4 + 1, i_5)$
  - ...

State-set encoding using MDDs

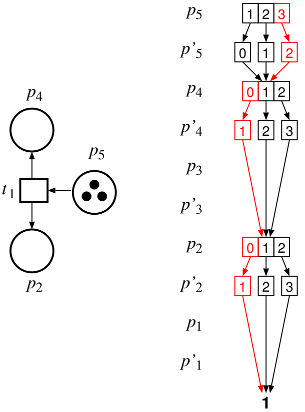
- MDD ordered over sequence of state variables $(u_L, \ldots, u_1)$
- Node m of MDD:
  - *Terminal* node : **0** and **1**, associated variable u_0
  - *Non-Terminal* node : associated variable u_k, $\forall i_k \in \mathcal{D}(u_k)$, an edge to child $m[i_k]$



Initial state $\mathbf{i}_0$ Reachable states $\mathcal{S}$

Transition encoding using MxDs

- Similar to MDD
- Except, *non-terminal* node m : associated variable u_k, $\forall(i_k, j_k) \in \mathcal{D}(u_k) \times \mathcal{D}(u_k)$, an edge to a child $m[i_k, j_k]$



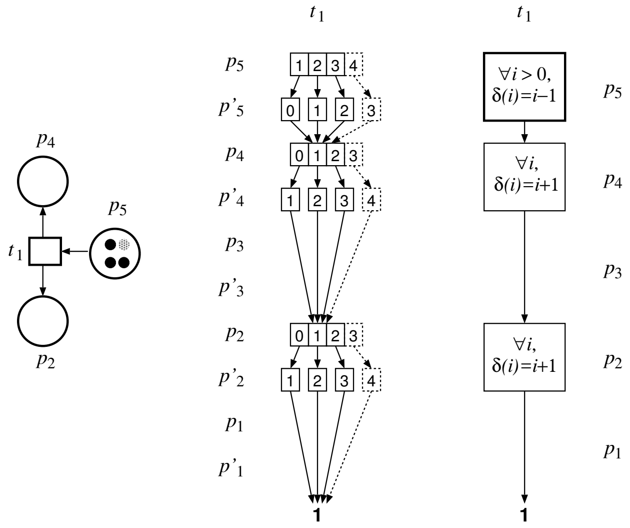
Saturation using MDDs & MxDs

- *Saturation* : Generates reachability set
  - Explores state-space in a bottom-up fashion.
  - Fires events :
    - $\mathcal{E}_1 =$ *Events associated with variables at level* 1
    - $\mathcal{E}_2 =$ *Events associated with variables at level* < 2
    - ...
    - $\mathcal{E}_L =$ *Events associated with variables at level* $< L$
  - Node saturation via persistent relational product operation
  - Iteration until fixed-point $\mathcal{S} = \{\mathbf{i}_0\} \cup \mathcal{S} \cup \mathcal{N}(\mathcal{S})$
- Domain of state variables during saturation
  - *Known* bounds: Guessing bounds not always possible; Static $\mathcal{N}$
  - *Unknown* bounds: "on-the-fly" saturation; Dynamic $\mathcal{N}$
- Extensible MDD/MxD nodes to efficiently handle growing domains during saturation.
- Overhead of operations to rebuild & update MxD nodes

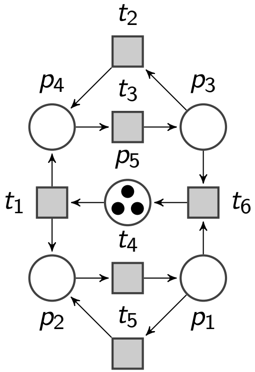
Implicit Relation Forest

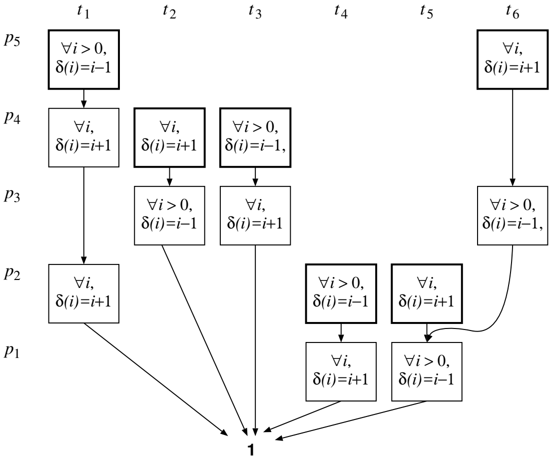
- Ordered, DAG
- Consists of $|\mathcal{E}|$ implicit relations
  - Implicit relation identifier : top-most node of each event
  - Relation node r of an implicit relation for e:
    - *Terminal* node : $\mathbf{1}$, associated variable u_0
      Non-terminal node : associated variable u_k, encoding function $r.\delta : \mathcal{D}(u_k) \nrightarrow \mathcal{D}(u_k) \equiv \Delta_{e,k}$
    - Has single outgoing edge to child $r.ptr$
    - Node identifier : $(u_k, r.ptr, r.\delta)$

Transition encoding using Implicit Relations



Transition encoding using Implicit Relations





Related Work

Existing alternative approaches of encoding transitions and their comparison with implicit relations :

- MxDs and extensible MxDs

MxDs [Clark et.al][M.Chung et.al]

Extensible MxDs address the issue of deletion of relevant yet incomplete compute-table entries in MxDs that reduce efficiency. However, Extensible MxDs have an overhead cost of node rebuilding during saturation process.

Related Work

Existing alternative approaches of encoding transitions and their comparison with implicit relations :

- MxDs and extensible MxDs
- Interval Mapping Diagrams

IMDs [Strehl et.al]

IMDs encode *state distance* between pre- and post-transition state variable values via use of *action operator* and *action interval* to determine the net-effect of the transition on *predicate interval*. The action operators are restricted to increment, decrement and equality operations only.

Related Work

Existing alternative approaches of encoding transitions and their comparison with implicit relations :

- MxDs and extensible MxDs
- Interval Mapping Diagrams
- Homomorphisms

Homomorphisms [Couvreur et.al]

Transitions are encoded using concept of inductive homorphisms which is defined to work with Data Decision Diagrams (DDD) and Hierarchical Set Decision Diagrams (SDD). The approach offers freedom of defining transitions to the user and is more efficient compared to prior works.

Experimental Setup

- Evaluation of state-space generation process :
  - Data-structure efficiency : OTFSAT vs SATIMP in SMART/Meddly (time & memory)
  - Benchmark Assessment : SMART vs ITS-Tools
- Suite of 70 Petri net models available as *known-models* in MCC 2018
- Experimental run timeout is set to 1 hour on Intel Xeon CPU 2.13GHz with 48G RAM under Linux Kernel 4.9.9

OtfSat vs SatImp

Model	$\lvert\mathcal{S}\rvert$	Runtime (sec)		Additional MxD computations in OtfSat	
		OtfSat	ImpSat	Pings ($\times 10^3$)	Hits ($\times 10^3$)
Safe Nets:					
DES 30a	1.92×10^{13}	22.60	23.91	24	16
DES 30b	1.97×10^{22}	103.97	102.74	18	11
FlexibleBarrier 10a	6.91×10^{10}	3.01	2.96	27	12
FlexibleBarrier 12a	8.92×10^{12}	15.75	15.74	219	10
Raft 5	5.94×10^{18}	23.58	23.91	11	7
Raft 6	2.91×10^{26}	189.464	193.92	18	12
RWmutex r10w100	1.12×10^{3}	2.29	2.48	102	69
RWmutex r10w500	1.52×10^{3}	51.18	40.15	586	422
Non-Safe Nets:					
FMS 100	2.70×10^{21}	8.90	4.12	341	50
FMS 200	1.95×10^{25}	78.28	33.50	50	50
GPPP C1000N10	1.42×10^{10}	1.19	0.21	43	43
GPPP C1000N100	1.14×10^{15}	440.35	114.88	18072	18071
Kanban 500	7.09×10^{26}	458.66	12.14	12704	12703
Kanban 1000	1.42×10^{30}	2347.18	72.50	50677	50436
Robot Manipulation 20	4.11×10^{9}	6.91	1.22	18	17
Robot Manipulation 50	8.53×10^{12}	176.05	33.15	253	253
SmallOS MT1024DC256	3.27×10^{12}	971.77	66.13	24522	24521
SmallOS MT2048DC0512	1.04×10^{14}	–	620.61	–	–
SmallOS MT2048DC1024	2.46×10^{14}	–	1105.18	–	–
SwimmingPool 9	1.81×10^{10}	11.73	7.28	70	70
SwimmingPool 10	3.36×10^{10}	15.81	10.98	95	95

OtfSat vs SatImp

Model	Memory usage (KB)	
	MxD	Implicit Relations
DES 40b	524.00	7.55
DNAwalker 15ringRRLarge	168.93	5.84
Angiogenesis 15	1,133.90	9.28
CircadianClock 1000	959,318.00	12.00
FMS 200	10,175.00	26.59
GPPP 100 100	1,470,591.00	48.56
Kanban 1000	464,342.00	16.00
Robot Manipulation 50	18,020.00	14.12
Small OS 1024 256	242,091.00	106.50
Swimming Pool 10	1,823.60	10.88

ITS-Tools as Benchmark for SMART:SatImp

Model	$\|\mathcal{S}\|$	Runtime (sec)		Approx. SI-Ratio
		SMART	ITS-Tools	
Kanban 100	1.7263E+19	1.42E+01	3.37E+03	1:238
SwimmingPool 6	1.6974E+09	2.89E+00	5.26E+01	1:18
Philosophers 500	3.6300E+238	2.24E-01	4.03E+00	1:18
HouseConstruction 10	1.6636E+09	5.99E-01	6.11E+00	1:10
ClientsAndServers 5	1.2551E+11	8.55E+00	7.11E+01	1:8
FMS 100	2.7031E+21	8.33E+00	6.31E+01	1:8
CircadianClock 100	4.2040E+10	4.60E-01	1.74E+00	1:4
IBMB2S565S3960	1.5511E+16	8.65E+00	1.60E+01	1:2
Ring	9.0265E+11	9.34E-02	1.72E-01	1:2
TokenRing 15	3.5358E+07	1.26E+01	1.57E+01	1:1.2
Referendum 100	5.1537E+47	8.96E-01	1.07E+00	1:1.2
SharedMemory 20	4.4515E+11	7.76E+00	5.04E+00	1.5:1
EnergyBus	2.1318E+12	9.00E+01	5.34E+01	1.69:1
Angiogenesis 5	4.2735E+07	9.58E-01	5.20E-01	1.8:1
FlexibleBarrier 4a	2.0737E+04	1.51E-01	6.11E-02	2.5:1
Railroad 10	2.0382E+06	5.94E+00	2.34E+00	2.5:1
Peterson 3	3.4079E+06	7.45E+01	2.70E+01	2.8:1
CSRepetitions 3	1.3407E+08	2.47E+00	6.01E-01	4:1
UtahNoC	4.7599E+09	3.44E+01	5.29E+00	6.5:1
PaceMaker	3.6803E+17	3.07E+00	2.47E-01	12.4:1

Conclusion

- Encode the functional effect from discrete-state model events instead of mapping-based representation.
- Shows increased efficiency of saturation algorithm in terms of time and memory.
- Not adapted to handle events with inter-variable dependency.

Future Work

- Intend to modify the implicit relations to represent more generic discrete-state models, like PNs w/ marking-dependent arc cardinalities.
- Inclusion of relation nodes into the decision-diagram (MxD)
  - Encode model events that are not Kronecker-consistent.
  - Retain the efficiency of encoding Kronecker-consistent events.

Questions?


IF SOMEONE SEES YOU, THEY'RE GONNA START ASKING QUESTIONS... QUESTIONS I DON'T HAVE ANSWERS TO!