Comparing CLP(FD) and SMT Approaches Applied to Workflow Nets Verification

H. Bride   O. Kouchnarenko   F. Peureux   **G. Voiron**

Département Informatique des Systèmes Complexes
Université de Bourgogne Franche-Comté

AFADL - June 7, 2016

Outline

Outline

Definition: Petri Net

A Petri Net is a tuple (P, T, F) where:

- P is a finite set of places
- T is a finite set of transitions ($P \cap T = \emptyset$)
- $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs

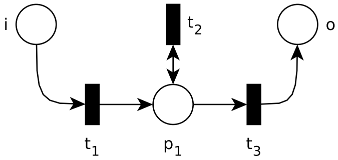


Figure 1 : A Petri Net

Petri Nets - Notations

Notations

Let $n \in P \cup T$ and $N \subseteq P \cup T$:

- ${}^{\bullet}n = \{n' \mid (n', n) \in F\}$
- $n^{\bullet} = \{n' \mid (n, n') \in F\}$

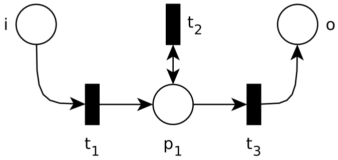


Figure 2 : A Petri Net

Examples

For the Petri Net in figure 2:

- ${}^{\bullet}p1 = \{t1, t2\}$
- $t2^{\bullet} = \{p1\}$

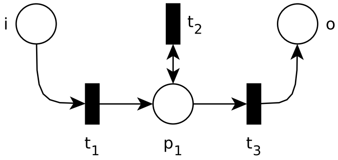


Figure 2 : A Petri Net

Execution of a Petri Net

Marking

- A marking of a Petri Net is a function $M : P \to \mathbb{N}$.
- A transition $t \in T$ is enabled if and only if $\forall p \in {}^{\bullet}t, M(p) \geq 1$.
- A transition can be fired if and only if it is enabled.
- A fired transition t modifies the marking of the Petri Net by:
  - Consuming one token from each place of ${}^{\bullet}t$
  - Producing one token for each place of $t^{\bullet}$

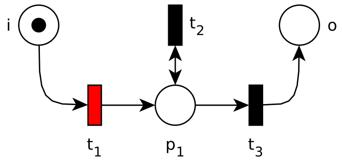


Figure 3 : Initial marking of a Petri Net

Execution of a Petri Net

Example

The following sequence of execution σ is valid for the Petri Net in figure 3:

$$\sigma = t1$$

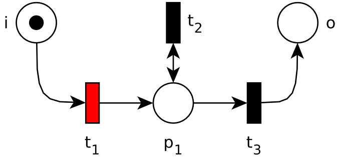


Figure 3 : Execution of a Petri Net

Execution of a Petri Net

Example

The following sequence of execution σ is valid for the Petri Net in figure 3:

$$\sigma = t1$$

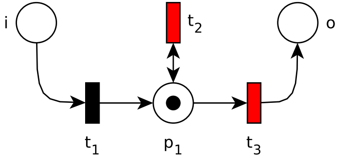


Figure 3 : Execution of a Petri Net

Execution of a Petri Net

Example

The following sequence of execution σ is valid for the Petri Net in figure 3:

$$\sigma = t1, t2$$

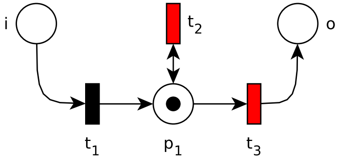


Figure 3 : Execution of a Petri Net

Execution of a Petri Net

Example

The following sequence of execution σ is valid for the Petri Net in figure 3:

$$\sigma = t1, t2, t2$$

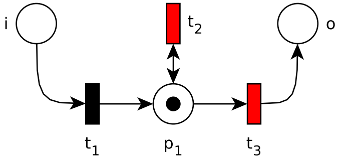


Figure 3 : Execution of a Petri Net

Execution of a Petri Net

Example

The following sequence of execution σ is valid for the Petri Net in figure 3:

$$\sigma = t1, t2, t2, t3$$

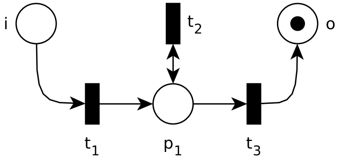


Figure 3 : Execution of a Petri Net

Definition: Workflow Net

A Petri Net PN = (P, T, F) is a Workflow Net if and only if:

- PN has two special places *i* and *o* where:
  - ${}^\bullet i = \emptyset$
  - $o^\bullet = \emptyset$
- For each node $n \in P \cup T$, there exists a path from *i* to *o* passing through *n*.

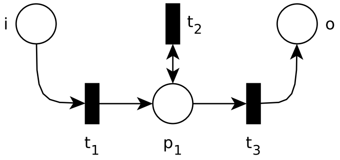


Figure 4 : A Workflow Net

Outline

Extended Modal Specifications

Definition

Let S be the language of well-formed modal specification formula:

- $\forall t \in T$, t is a well-formed modal formula.
- Given A_1, $A_2 \in S$, $A_1 \wedge A_2$, $A_1 \vee A_2$, and $\neg A_1$ well-formed modal formula.

Extended Modal Specifications

- Express requirements on several transition and on their causalities.
- A modal specification formula $m \in S$ can be interpreted as:
  - a *may*-formula - a behaviour that has to be ensured by **at least one** correct execution
  - a *must*-formula - a behaviour that has to be ensured by **all** correct executions

Example of Modal Specifications

- $PN \models_{may} \neg t2 \wedge t3$ is **valid** for the Workflow Net in figure 5.
- $PN \models_{may} t1 \wedge \neg t3$ is **invalid** for the Workflow Net in figure 5.
- $PN \models_{must} t1 \wedge t3$ is **valid** for the Workflow Net in figure 5.
- $PN \models_{must} t1 \wedge t2$ is **invalid** for the Workflow Net in figure 5.

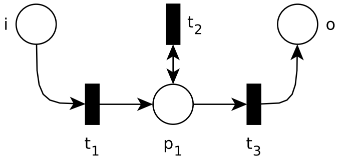


Figure 5 : A Workflow Net PN

Outline

Introduction - CLP(FD) and SMT

Constraint Logic Programming Over Finite Domains

- Can solve Constraint Satisfaction Problems
- Constraints on variables with finite domains
- Search space exploration with backtracking
- SICStus Prolog has been chosen

Satisfiability Modulo Theory

- Can solve Constraint Satisfaction Problems
- Constraints on variables with infinite domains
- Combination of a SAT-solver and a Theory-solver
- Z3 has been chosen

Outline

Verification Method - Process

Modal Specification Verification Processs

- Compute an over-approximation of the set of correct executions of the Workflow Net:
  - No execution invalidates the specification → Specification **valid**.
  - An execution invalidates the specification → Compute a correct execution invalidating the specification:
    - Such an execution exists → Specification **invalid**.
    - Otherwise, compute an other under-approximation until either the specification is violated or no correct execution violates the specification.

Outline

Verification Method - Constraints I

Fundamental State Equation

Computing an over-approximation of the executions of a Workflow Net:

$$\forall p \in P, \nu(p) = \sum_{t \in {}^{\bullet}p} \nu(t) + M_a(p) = \sum_{t \in p^{\bullet}} \nu(t) + M_b(p)$$

where $\nu : P \rightarrow \mathbb{N}$ is a valuation function.

Modal Formula

Verifying a modal formula f relies on its expression by constraints (denoted $C(f, \nu)$):

- For every transition $t \in T$, the corresponding terminal symbol of the formula is replaced by $\nu(t) > 0$, where ν is the valuation function.

Verification Method - Constraints II

Siphon detection

Avoiding deadlocks in the executions of the over-approximation:

$$\forall p \in P, \forall t \in {}^{\bullet}p. \sum_{p' \in {}^{\bullet}t} \xi(p') \geq \xi(p) \wedge \sum_{p \in P} \xi(p) > 0$$

Other Constraints

Computing correct executions (under-approximation) of the Workflow Net. For further information about these constraints, the interested audience can read [BKP14].

Outline

Data Set Generation Process

Process

1. Define modal specification configuration (type, size and logical operators used)
2. Derive the formula from its configuration
3. The modal formula is generated
4. Generate a Workflow Net from the formula
5. A minimal Workflow Net is generated
6. Expand the size of the Workflow Net
7. The real Workflow Net is generated

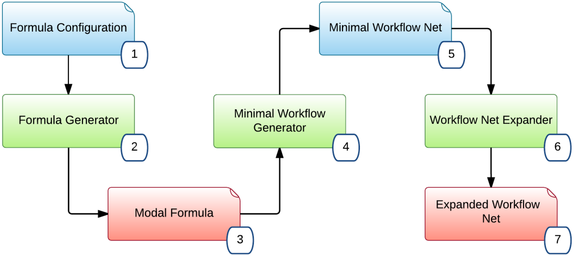


Figure 6 : Data Set Generation Process

Outline

Verification Toolchain Architecture

Process

1. Take the Workflow Net and the Modal Specification as input
2. Generate verification code for both Z3 and SICStus
3. Communicate with solvers to determine the validity of the modal specification
4. Generate a report about the validity of the specification:
   - Valid / Invalid / Timeout?
   - Verification time
   - Number of segments

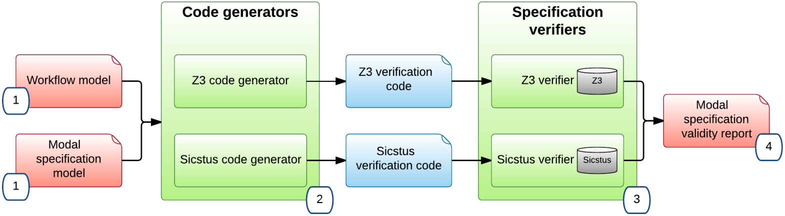


Figure 7 : Verification Process

Outline

Experimental Protocol I

Type of modal specification

The four types of modal specifications have been generated and verified:

- Valid may-formula
- Invalid may-formula
- Valid must-formula
- Invalid must-formula

Size of the modal formula

Two sizes for the modal specifications have been used:

- Specifications using 5 literals
- Specifications using 15 literals

Experimental Protocol II

Class of the workflow nets

Different classes of workflows have been generated and verified:

- State machine (allowing conflicts)
- Marked graph (allowing concurrency)
- Free-choice (allowing conflicts and concurrency, not in the same time)
- Ordinary nets

Size of the workflow nets

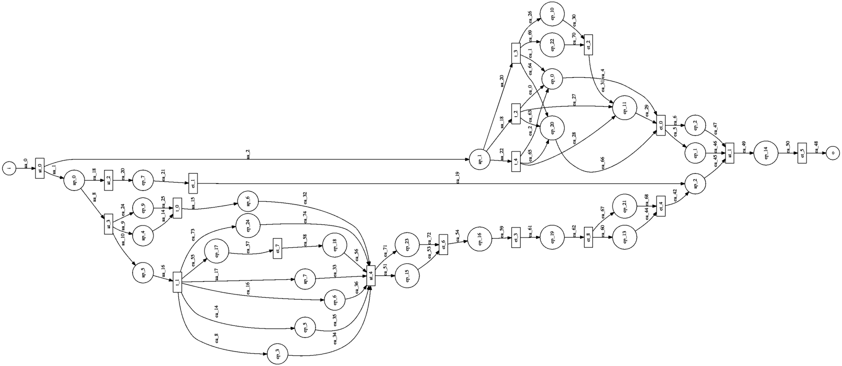
Workflows of growing size have been generated and verified:

- 50 nodes
- 100 nodes
- ...
- 500 nodes

Experimental Protocol III

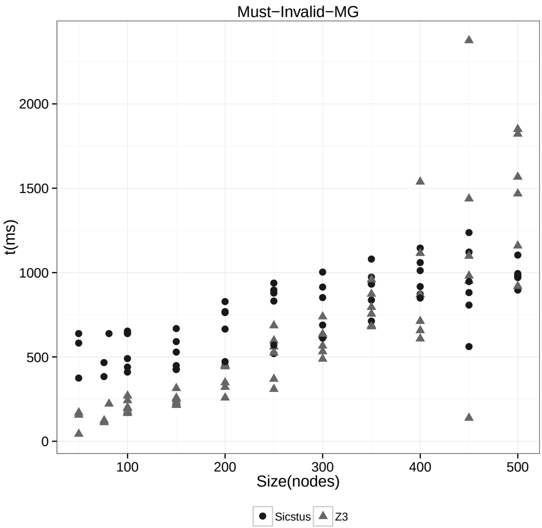
Complete Data Set

- 3 iterations
- 320 instances of growing size and complexity per iteration
- Total: 960 Workflow Nets and Modal Specifications verified



Outline

Must invalid Specifications Verification (Marked-Graphs)



Metrics over Marked-Graph Workflow Nets

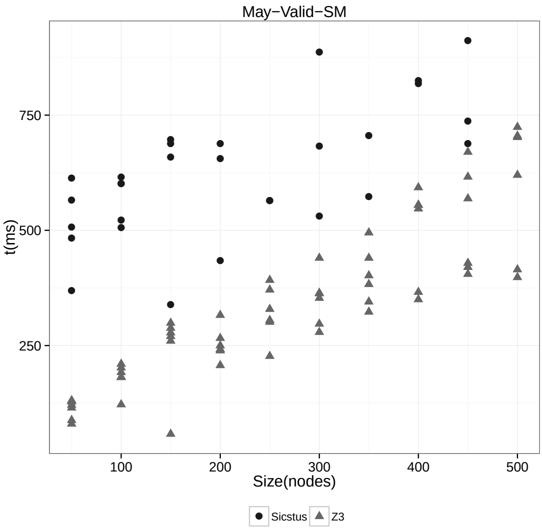
Table 1 : Metrics over Marked-Graph Workflow Nets

Type	Solver	Avg. t.(ms)	#time-outs	Overall
May-Valid	Z3	630	0	☺
	SICStus	776	0	☺
Must-Invalid	Z3	641	0	☺
	SICStus	758	0	☺
May-Invalid	Z3	112	0	☺
	SICStus	424	0	☺
Must-Valid	Z3	104	0	☺
	SICStus	407	0	☺

Synthesis

- Both Z3 and SICStus can handle Marked-Graphs efficiently.

May Valid Specifications Verification (State-Machines)



Metrics over State-Machine Workflow Nets

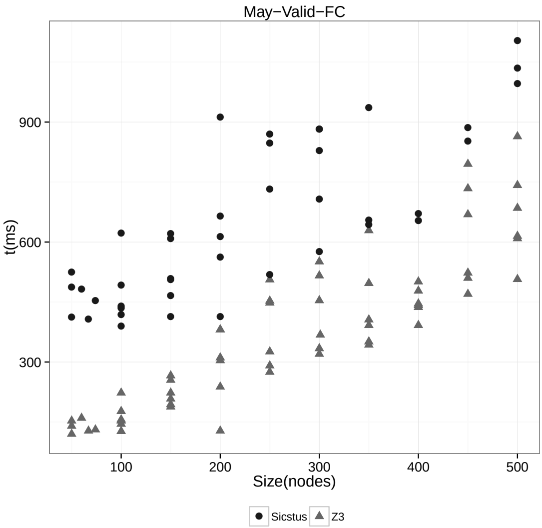
Table 2 : Metrics over State-Machine Workflow Nets

Type	Solver	Avg. t.(ms)	#time-outs	Overall
May-Valid	Z3	346	0	🙂
	SICStus	621	28	😐
Must-Invalid	Z3	319	0	🙂
	SICStus	788	31	☹️
May-Invalid	Z3	79	0	🙂
	SICStus	77413	52	☹️
Must-Valid	Z3	79	0	🙂
	SICStus	10194	51	☹️

Synthesis

- Z3: No time-out and small average verification time.
- SICStus: Many time-outs and high average verification time.

May Valid Specifications Verification (Free-Choice Nets)



Metrics over Free-Choice Workflow Nets

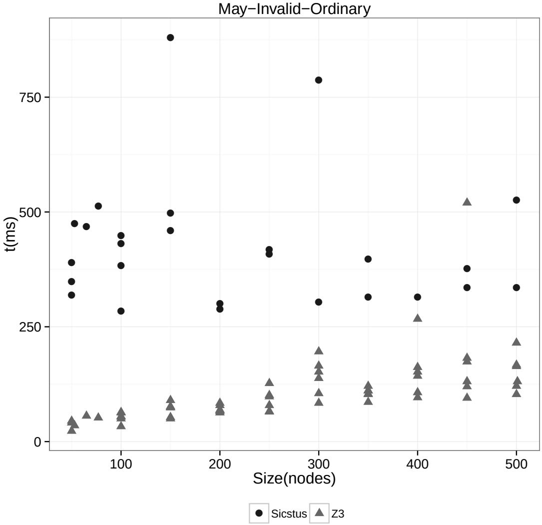
Table 3 : Metrics over Free-Choice Workflow Nets

Type	Solver	Avg. t.(ms)	#time-outs	Overall
May-Valid	Z3	379	0	
	SICStus	787	16	
Must-Invalid	Z3	413	0	
	SICStus	898	14	
May-Invalid	Z3	91	0	
	SICStus	40459	38	
Must-Valid	Z3	89	0	
	SICStus	50566	37	

Synthesis

- Z3: No time-out and small average verification times.
- SICStus: Many time-outs and high average verification times.

May Invalid Specifications Verification (Ordinary Nets)



Metrics over Ordinary Workflow Nets

Table 4 : Metrics over Ordinary Workflow Nets

Type	Solver	Avg. t.(ms)	#time-outs	Overall
May-Valid	Z3	1258	22	
	SICStus	9010	33	
Must-Invalid	Z3	713	17	
	SICStus	12258	37	
May-Invalid	Z3	108	0	
	SICStus	9489	33	
Must-Valid	Z3	106	0	
	SICStus	5949	37	

Synthesis

- Z3: many time-outs with *may*-valid and *must*-invalid specifications.
- SICStus: many time-outs and high average resolution time.

Outline

Conclusion

Feedback

- The verification method's scalability has been demonstrated.
- Z3: Performs well and better than SICStus on most examples.
- SICStus: Difficulties with most classes, performs well with Marked Graphs.

Perspectives

- Refine constraints to improve verification time and avoid time-outs.
- Mixing SMT and CLP approaches to embrace the benefits from each of them.
- Validate the efficiency of the method against real life case studies.

Hadrien Bride, Olga Kouchnarenko, and Fabien Peureux, *Verifying modal workflow specifications using constraint solving*, Integrated Formal Methods, Springer, 2014, pp. 171–186.

Thank you for your attention.

Any questions?