JANI2PINS

The JANI front-end module for LTSmin

Lars Schieffer

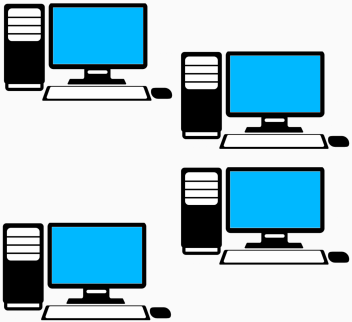
Thursday 15th October, 2020

Outline

- Various languages around the world
  - Possibility 1: **Translator**
  - Possibility 2: **Common language**

Picture Source: https://www.freepik.com/free-vector/people-speaking-avatar-collection_1116732.htm

Outline



- Various languages around the world
  - Possibility 1: **Translator**
  - Possibility 2: **Common language**
- Model checkers: Various formalisms
  - Translator: Unreasonable task
  - Common formalism: **JANI**

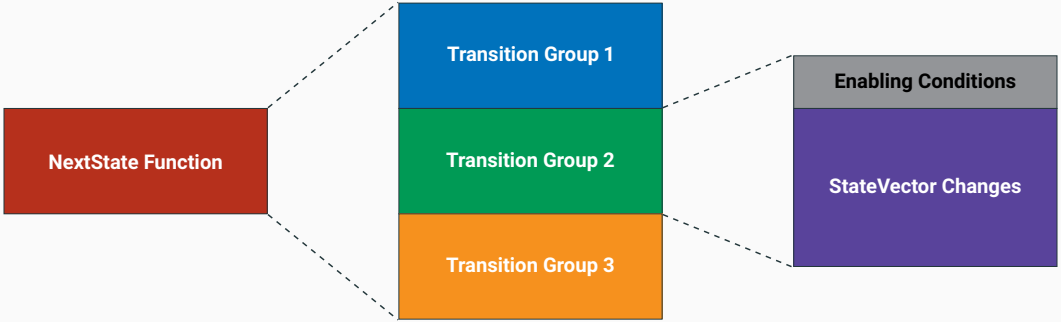
Picture Source: https://www.clipartmax.com/middle/m2i8H7b1H7G6i8H7_computer-icon-vector-png-computer-science-clipart-black-and-white/

The LTSmin Toolset

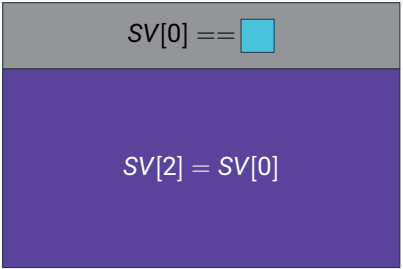
The LTSmin Toolset

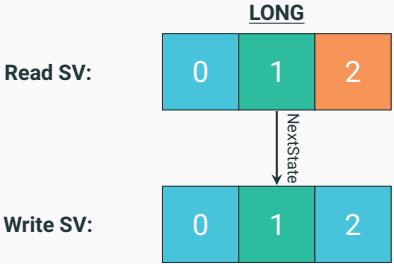
- LTL/CTL/μ-calculus model checker
- Model checking algorithms are performed on *Partitioned Transition Systems*:
  - $S_{\mathcal{P}} = S_1 \times \ldots \times S_N$ as state space, containing all possible state vectors with N slots
  - $\rightarrow_{\mathcal{P}} = \bigcup_{i=1}^{K} \rightarrow_i$ as labelled transition relations, described by K transition groups
  - $\rightarrow_i \subseteq S_{\mathcal{P}} \times \mathcal{A} \times S_{\mathcal{P}}$ as transition group
  - $s^0 = \langle s_1^0, \ldots, s_N^0 \rangle$ as initial state vector
  - $L : S_{\mathcal{P}} \times \mathcal{L} \rightarrow \mathbb{N}$ as state labelling function
- **Advantage**: Break transition system into smaller independent parts
- **P**artitioned **I**nterface for **N**ext **S**tates (PINS) models
- Add new language modules as compiled libraries (dl-open API)

Transition Relations: NextState Function



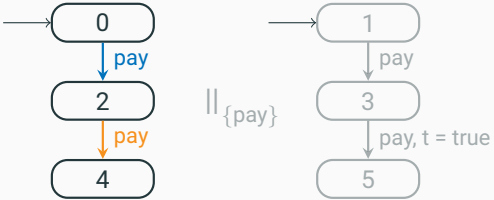
StateVector Variations





Creation of Transition Groups

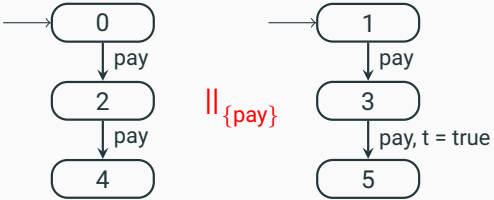
Global variables:
bool t = false;



- Every Edge ⇒ Transition Group

Creation of Transition Groups

Global variables:
bool t = false;

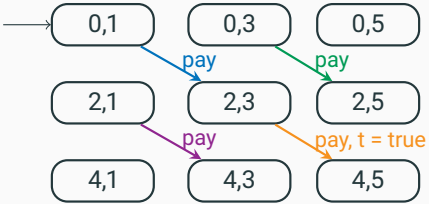


- Every Edge ⇒ Transition Group
- Parallel Composition:
  - Not available in PINS models
  - **Requires:** Implementation inside transition groups

Creation of Transition Groups

Global variables:

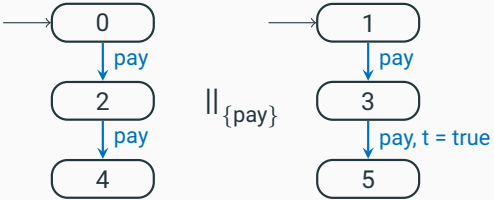
bool t = false;



- Every Edge ⇒ Transition Group
- Parallel Composition:
  - Not available in PINS models
  - **Requires:** Implementation inside transition groups
- Multiple Solutions:
  - Synchronous Product

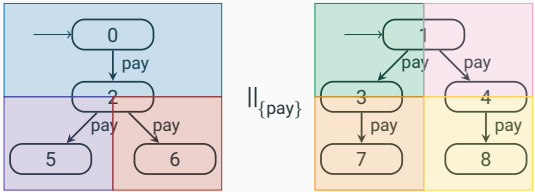
Creation of Transition Groups

Global variables:
bool t = false;



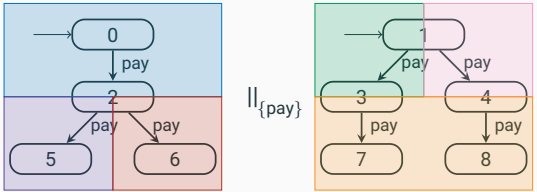
- Every Edge ⇒ Transition Group
- Parallel Composition:
  - Not available in PINS models
  - **Requires:** Implementation inside transition groups
- Multiple Solutions:
  - Synchronous Product
  - Symbolic Approach

Symbolic Approach with Nondeterminism



- Symbolic Transition Groups:
  - Limited to deterministic edges
    $\Rightarrow$ Dissolve ND-Choices inside transition groups
- **Solution 1**: Synchronous Product
  - Creates $left * right = 12$ TGs

Symbolic Approach with Nondeterminism



- Symbolic Transition Groups:
  - Limited to deterministic edges
    $\Rightarrow$ Dissolve ND-Choices inside transition groups
- **Solution 1**: Synchronous Product
  - Creates $left * right = 12$ TGs
- **Solution 2**: Symbolic Product
  - Creates $left * right = 9$ TGs

Effects of Grouping Strategies

Optimisation Power of LTSmin

FLATTEN	MIXED	SYMBOLIC
Synchronous Product on origin edges	Synchronous Product on origin edges	Synchronous Product on grouped edges
✚ Maximum freedom for LTSmin ✖ Explosion of transition groups	✚ Merge of both worlds ✖ Overrule LTSmin algorithms	✚ Inhibits transition groups explosion ✖ Overrule LTSmin algorithms

JANI Specification

General Overview

- Model Format:
  - Properties
  - Automata definitions
  - Automata composition
  - Dataformat: JSON
- Protocol (WebSocket):
  - Purpose: Model exchange
  - Roles: Transformation & Analysis

```
{
  "jani-version":1,
  "name":"correctness",
  "type":"lts",
  "features":[
    "derived-operators"
  ],
  "actions":[
    {
      "name":"a"
    }
  ],
  "properties":[
    {
      "name":"P",
      "expression":{
        "op":"filter",
        "fun":"max",...
```

Properties

(✖) PINS models do not contain properties

(✖) LTSmin restricted to LTL, CTL or μ-calculus properties

(✚) Linear Time Property:

jani-model:	Pmin(<> variable)
	↓
LTSmin:	<> variable

(✚) Branching Time Property: Choice between $\exists$ or $\forall$ paths

Model Types

Type	(✚) LTS	(✖) TA
Transformation	(✚) Available	(✖) Clocks
Workaround		(✖) External Library Call
State Space	(✚) Consistent	
Type	(✖) DTMC	(✖/✚) MDP
Transformation	(✖) Probability Experiments	(✖) Probability Experiments
Workaround	(✚) Individual Edges	(✚) Individual Edges
State Space	(✖) ND Choices at States	(✖/✚) Consistent (expanded)

Possible transformations

- ✔ Automata instances composition:
  - Implementation of presented grouping strategies
- ✔ Input-enabled automata:
  - Transformation with explicit self-loop edges (inspired by moconv option)
- ✔ Transient and non-transient variables:
  - Variables: Slot in StateVector
  - Transients:
    - Value is reset for each NextState call and TG
    - Update value according actual automata locations
- ✔ Assignment ordering in local and synchronised edges:
  - Atomic steps are represented by intermediate local variables
- ✔ Function Extension: Create C source code
  - Recursive function possible, but
  - Prohibits direct access to local or global variables

Limitations of transformations

- Expressions:
  - ✖ Sampling of Probability Distributions
  - ✔ Bool-Type and Integer-Type
  - ✖ Real-Type
  - ✖ "/"-Operator: Always Real-Type Result
  - ✔ "%"-Operator: Euclid Remainder
- Array Extension:
  - ✖ Inside StateVector: Dynamic array sizes
  - ✖ C array: Values not in model state space

JANI2PINS

Implementation: JANI2PINS

- **Purpose 1**: Creation of single model language front-ends for LTSmin
  - Implementation of possible jani-model transformations
  - Provide several grouping strategy options to user
  - Initialisation of only declared jani-model constants
- **Purpose 2**: Server implementation of jani-interaction protocol
  - Analysis engine: LTSmin
  - Transformer: JANI2PINS transformations
- **Correctness**: Verifying process of transformations
  - State Space Check: Consistent state space after transformation
  - Property Check: Verify expected model behaviour

The Interaction Protocol

Interaction Protocol: PseuCo

pseuCo.com Files Backup Help About

Editing unsaved file CCS Duplicate

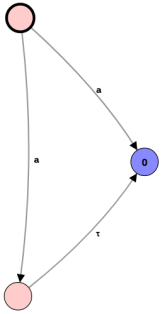
CCS → LTS Actions

CCS

```
(a.0 + a.i.0)
```

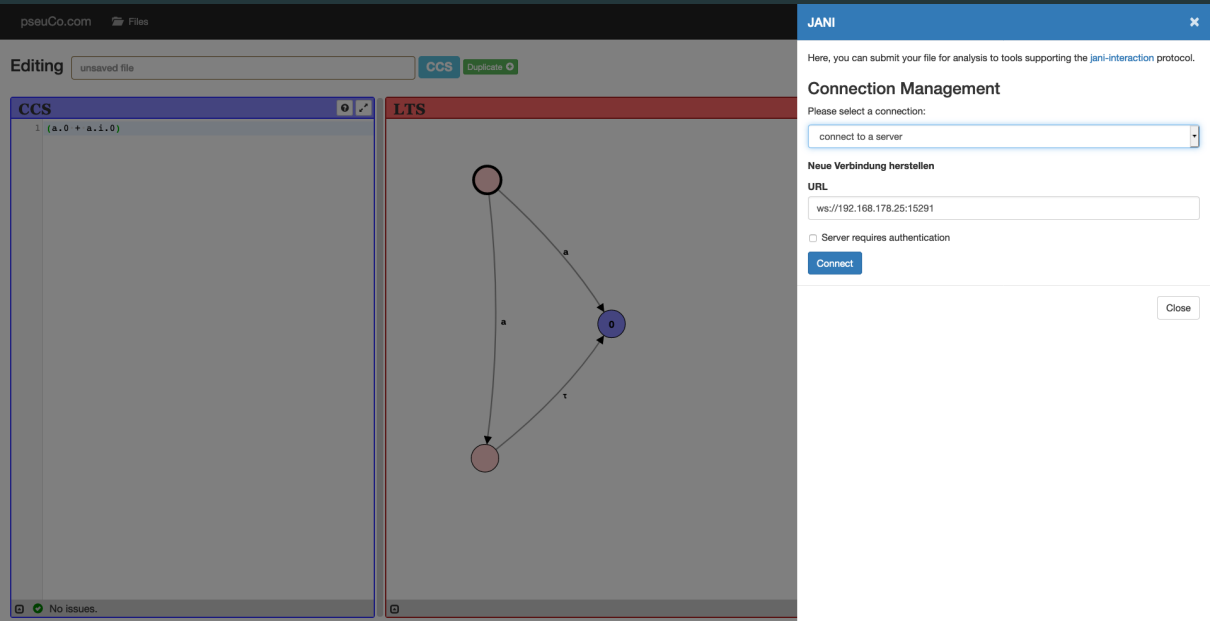
No issues.

LTS



Minimize Collapse all Expand all

Interaction Protocol: PseuCo



Interaction Protocol: PseuCo

pseuCo.com Files

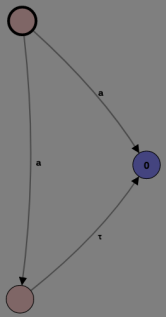
Editing unsaved file CCS Duplicate

CCS

```
1 (a.0 + a.i.0)
```

No issues.

LTS



Submit Analysis Job

Engine Selection

Select an analysis engine:

JANI2PINS - Analysis Engine

Select an input type:

lts (JANI)

The LTS needs to be exported before this task can be submitted. Only finite LTS are supported.

✔ The LTS has been exported.

Set Parameters

This engine has parameters that can be configured:

☐ **Real2Int**
Cast all real values to integer

Grouping Strategy
Set Grouping Strategy

FLATTEN

LTSmin Backend
Set backend for performing model checking

sequential

Command
Command line arguments for LTSmin

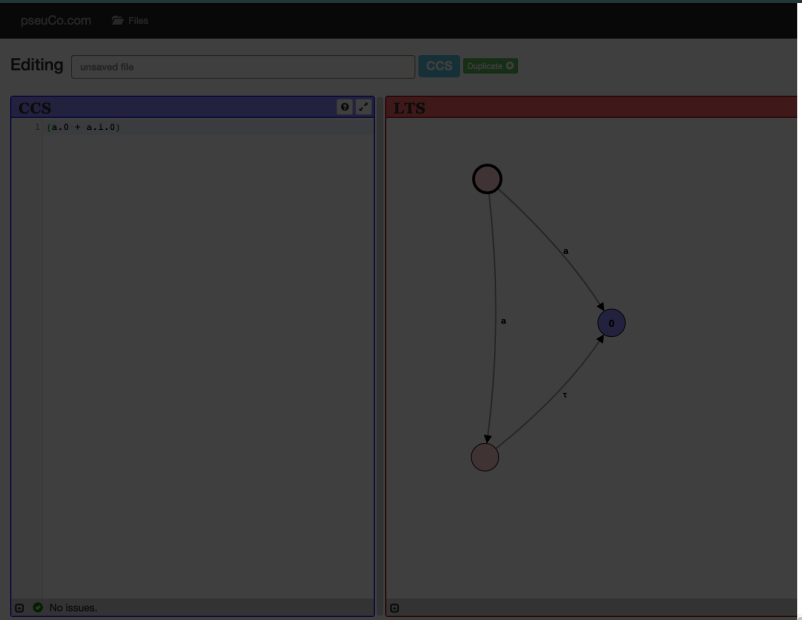
--deadlock

☑ **Trace**
Print trace, if available

Submission

Submit Analysis Task

Interaction Protocol: PseuCo

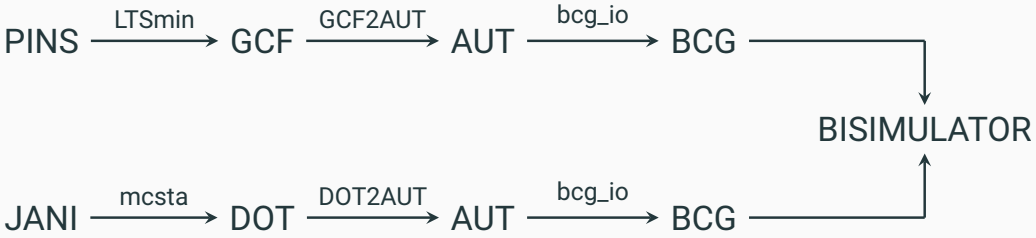


Results

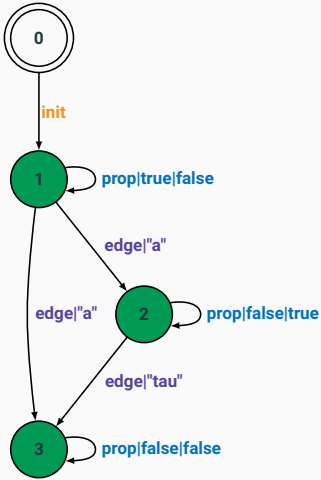
Label	Value
pins2lts-seq	Registering PINS so language module
pins2lts-seq	Loading model from /Users/lars/Cloud/Drive/Uni/Thesis/JANI2PINS /tmp/140509096262720/2/model.so
pins2lts-seq	library has no initializer
pins2lts-seq	loading model jani2pins
pins2lts-seq	completed loading model jani2pins
pins2lts-seq	There are 2 state labels and 1 edge labels
pins2lts-seq	State length is 1, there are 3 groups
pins2lts-seq	Running dfs search strategy
pins2lts-seq	Using a tree for state storage
pins2lts-seq	deadlock () found at depth 3!
pins2lts-seq	Writing trace to /Users/lars/Cloud/Drive/Uni/Thesis/JANI2PINS /tmp/140509096262720/2/traces/traces.gcf
pins2lts-seq	exiting now
Trace Output	----------
Action	a
Action	tau

Verifying Process

Verifying Process



State Space: PINS Model



1. Remove init transition
2. Remove StateLabel transitions
3. Decrease Locations
4. Update Edge Labels

GCF2AUT

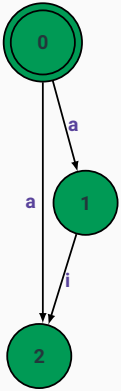
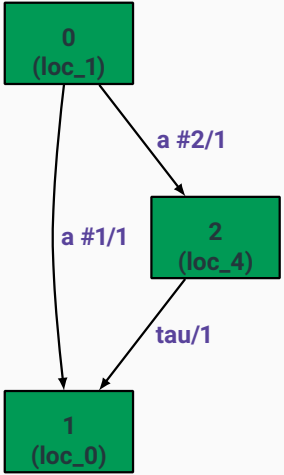


Figure 3: LTSmin state space for CCS term: (a.0 + a.τ.0)

State Space: JANI Model



1. Extract Locations
2. Update Edge Labels

DOT2AUT

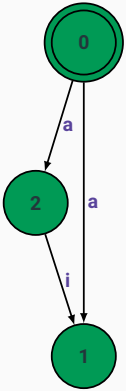
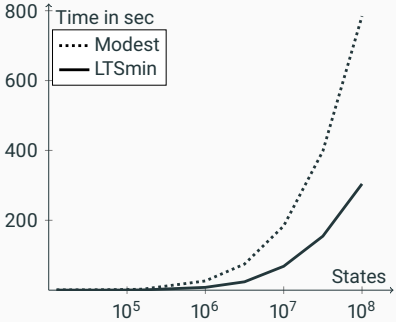


Figure 4: Modest (mcsta) state space for CCS term: (a.0 + a.τ.0)

Verifying Results

- **Goal**: State space and property check on sufficient amount of jani-models
- **Assistance**: QCOMP 2020 Benchmark Set
  - Variety of jani-models from third party
  - Model sources from various origins (PRISM, Modest...)
- **Results**:
  - ✚ QCOMP MDP: 13/18 Models (10 without OOM)
  - ✚ Models which confirm specification characteristics



Conclusion

Conclusion: JANI2PINS

- Expansion of the JANI format to LTSmin toolset
- Full control over grouping strategies
- Facilitated model checking process by server implementation
- Support of more JANI model types?
  - Possible: YES! (C Source Code)
  - Reasonable: NO, since mainly designed for LTS and its property logic

[1] A. Hartmanns, M. Klauck, D. Parker, T. Quatmann, and E. Ruijters, “The quantitative verification benchmark set,” in *Tools and Algorithms for the Construction and Analysis of Systems*, T. Vojnar and L. Zhang, Eds., Cham: Springer International Publishing, 2019, pp. 344–350.

[2] C. E. Budde, C. Dehnert, E. M. Hahn, A. Hartmanns, S. Junges, and A. Turrini, “JANI: quantitative model and tool interaction,” in *Tools and Algorithms for the Construction and Analysis of Systems - 23rd International Conference, TACAS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings, Part II*, A. Legay and T. Margaria, Eds., ser. Lecture Notes in Computer Science, vol. 10206, 2017, pp. 151–168.

[3] A. Laarman, E. Pater, J. van de Pol, and H. Hansen, “Guard-based partial-order reduction,” English, *International journal on software tools for technology transfer*, vol. 18, no. 4, pp. 427–448, Aug. 2016, eemcs-eprint-23356.

[4] A. Hartmanns and H. Hermanns, “In the quantitative automata zoo,” *Sci. Comput. Program.*, vol. 112, pp. 3–23, 2015.

[5] G. Kant, A. Laarman, J. Meijer, J. van de Pol, S. Blom, and T. van Dijk, “LTSmin: High-performance language-independent model checking,” in *Tools and Algorithms for the Construction and Analysis of Systems*, C. Baier and C. Tinelli, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 692–707.

[6] A. Hartmanns and H. Hermanns, “The modest toolset: An integrated environment for quantitative modelling and verification,” in *Tools and Algorithms for the Construction and Analysis of Systems - 20th International Conference, TACAS 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014. Proceedings*, E. Ábrahám and K. Havelund, Eds., ser. Lecture Notes in Computer Science, vol. 8413, Springer, 2014, pp. 593–598.

[7] H. Garavel, F. Lang, R. Mateescu, and W. Serwe, “CADP 2011: A toolbox for the construction and analysis of distributed processes,” *International Journal on Software Tools for Technology Transfer*, vol. 15, no. 2, pp. 89–107, Apr. 2013.

[8] D. Bergamini, N. Descoubes, C. Joubert, and R. Mateescu, “Bisimulator: A modular tool for on-the-fly equivalence checking,” in *Tools and Algorithms for the Construction and Analysis of Systems*, N. Halbwachs and L. D. Zuck, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 581–585.