SysML Models Verification Relying on Dependency Graphs

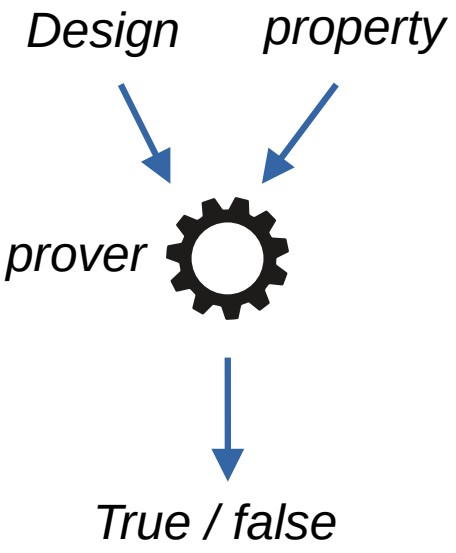
Ludovic APVRILLE
Pierre de SAQUI-SANNES
Oana HOTESCU
Alessandro TEMPIA CALVINO

Modelsward'2022

Context: Verifying Models

Model verification

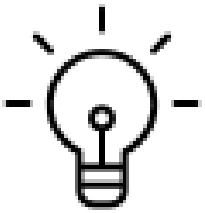
- Syntax checking
- Simulation
  - Random or user-guided
- Formal verification
  - Model transformed into a formal description
  - Use of a model-checker



Solutions to combinatory explosion

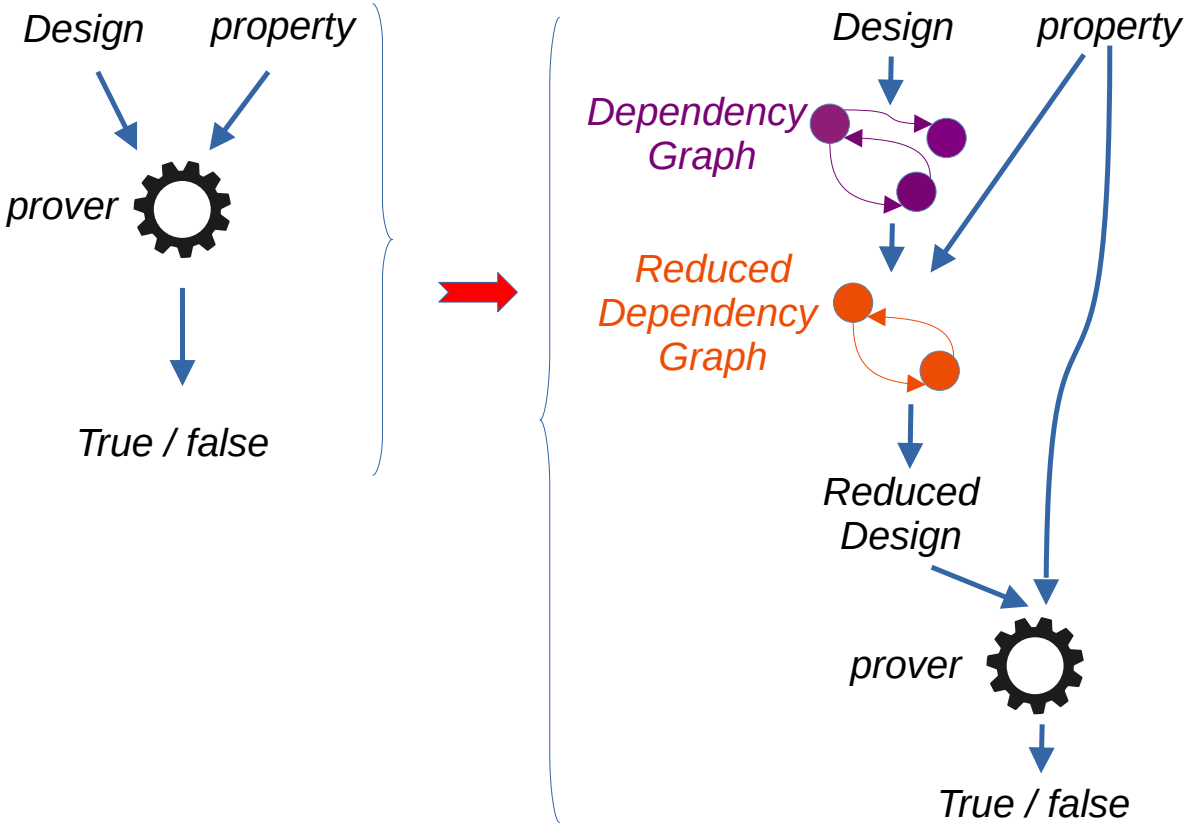
- Higher abstraction level
- Improving proofs

Automated Simplification of Models



Only the relevant part of a model is used to prove a property

→ Use of dependency graphs



Application to SysML Models and CTL Properties

SysML Model

- Structure: block definition diagrams and internal block diagrams
- Behavior : state machine diagrams (1 state machine per block)
- "SysML design" = Structure + Behavior

Properties

- Properties in CTL format
- $p = A|E \quad <> |[] \quad expr$ where *expr* is as follows:
  - Block.state
  - A boolean expression built upon blocks' attributes

Dependencies in SysML Models

Model elements	Dependency graph	Diagrams
State	One vertex per state	State machines
Transition	One edge per transition	State machines
Message sending / receiving (asynchronous)	One vertex per message sending/receiving. One edge between senders to all possible receivers.	State machines and internal block diagrams
Message sending / receiving (synchronous)	Like asynchronous, but double edge between potential couples sending / receiving actions	State machines and internal block diagrams

Implementation in TTool

Free and open-source toolkit supporting several SysML extensions (AVATAR, DIPLODOCUS, SysML-Sec).

1. Support of SysML design and CTL properties
2. Generation of dependency graphs from SysML designs
3. Generation of a reduced SysML Model from a dependency graph and one property
4. Internal model-checking from a (reduced) SysML design + one property [modelsward'2021]
5. Back-tracing to models

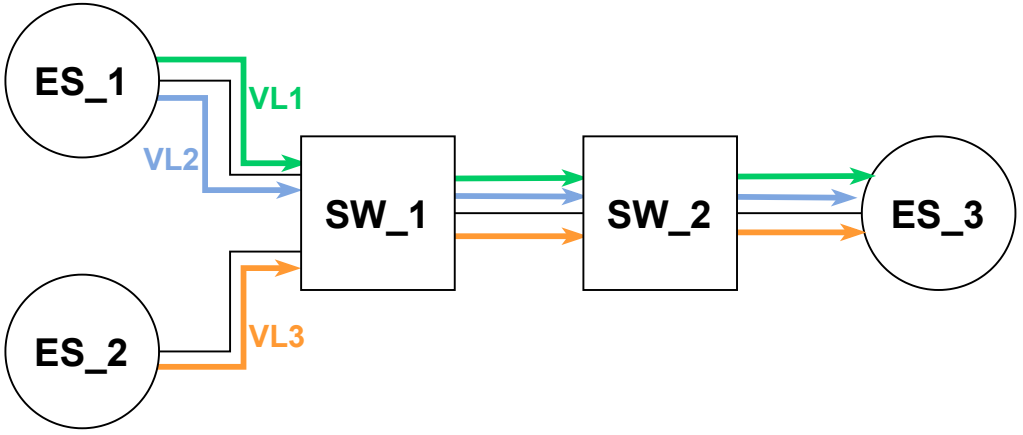
[modelsward'2021] A. Tempia Calvino, L. Apvrille, "Direct Model-Checking of SysML Models", Proceedings of Modelsward'2021, Vienna, Autrichia (online)

Case Study: An AFDX Network

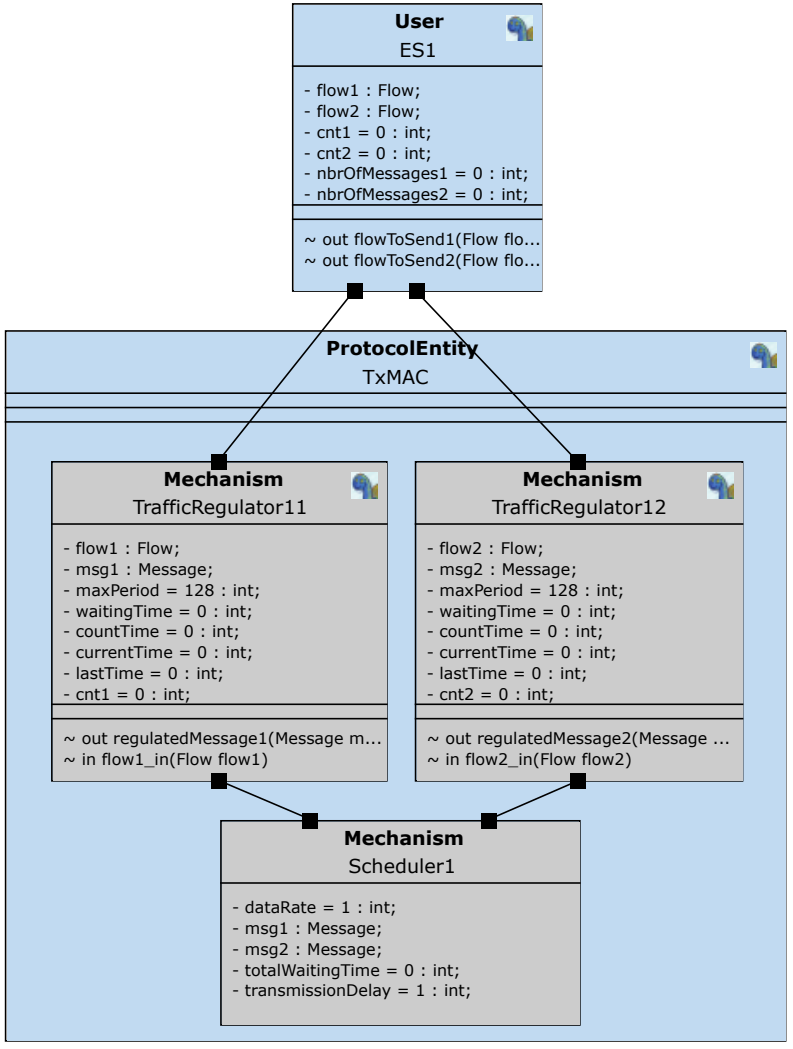
AFDX (Avionics Full DupleX switched Ethernet)

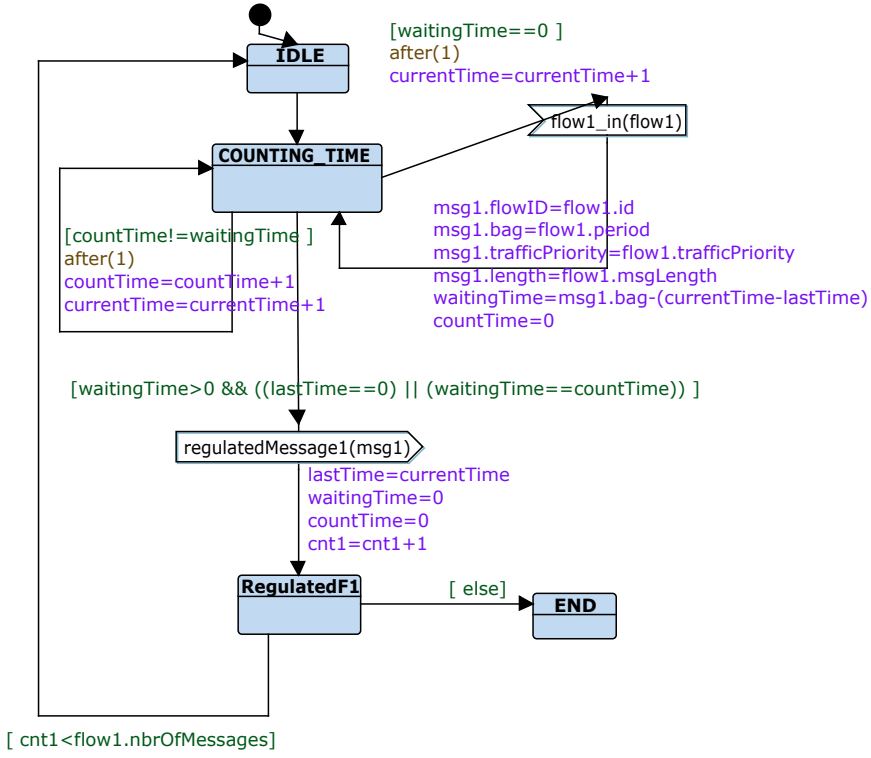
- De-facto embedded avionics network, (e.g. Airbus A380)
- Properties to be verified:
  - Message arrival time $<$ deadline
  - Total message ordering
  - No message loss due to buffer overflow or failure
- AFDX configuration (this paper)

ES: End System
SW: SWitch
VL: Virtual Link



SysML Model of the AFDX Network





State machine diagram of TrafficRegulator11

35 blocks and 125 states

Properties of the AFDX Network

Properties cover important network mechanisms (regulation, scheduling, filtering, demultiplexing)

Reachability

1. $E <> TrafficRegulator.RegulatedF3$
2. $E <> SWScheduler.ScheduledMsg1$
3. $E <> Filtering.FilteredF3$
4. $E <> Demultiplexer.ReceivedF2$

Liveness

1. $A[] TrafficRegulator.RegulatedF3$
2. $A[] SWScheduler.ScheduledMsg1$
3. $A[] Filtering.FilteredF3$
4. $A[] Demultiplexer.ReceivedF2$

Experiment

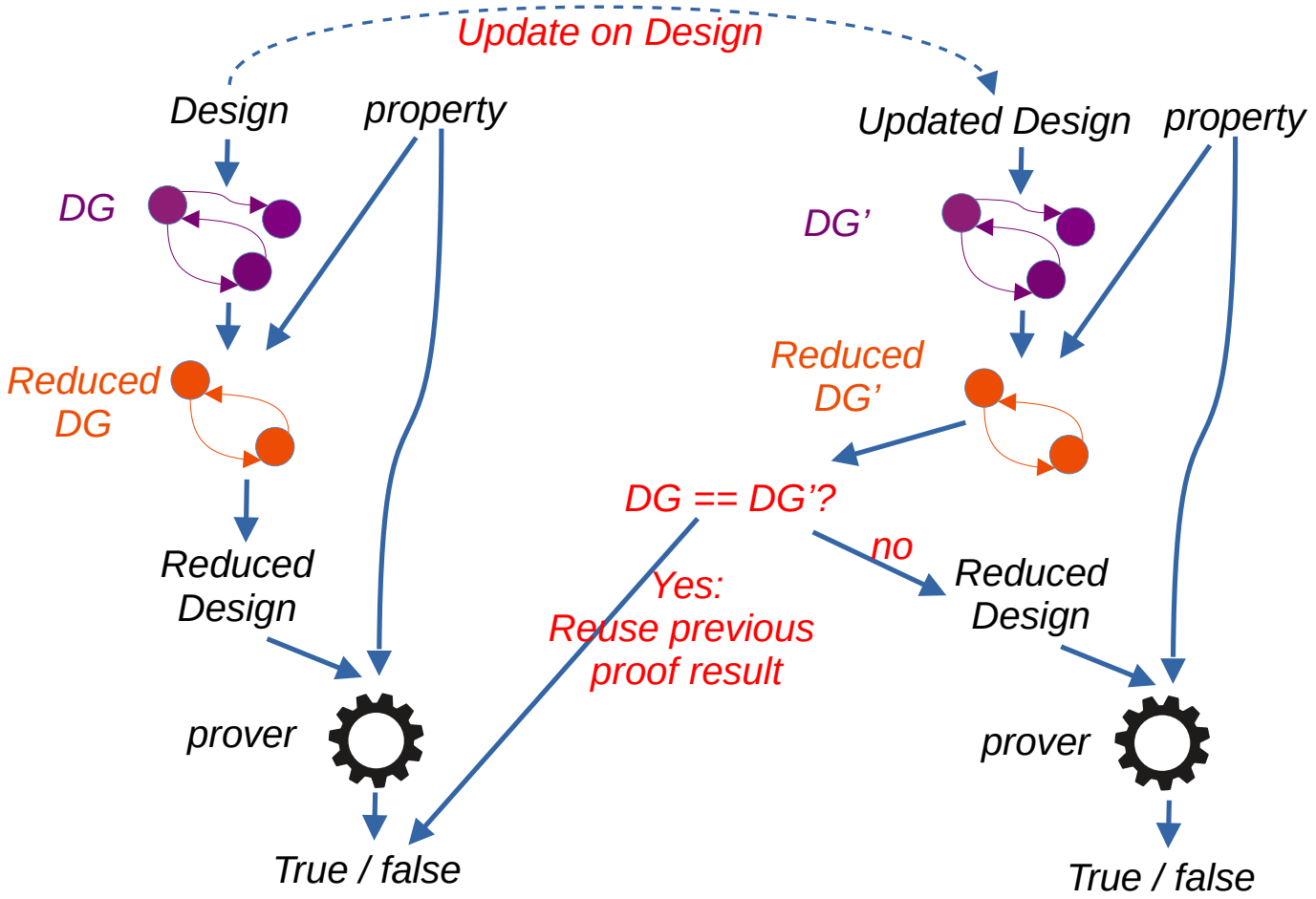
- Dependency graph size: 500 states, 600 transitions
- Reachability graph size: 165k states, 316k transitions

Property	Proof duration (ms)							Gain
	No DG			With DG				
	E <>	A[]	Total	E <>	A[]	DG processing	Total	
1	6	50	56	6	8	15	29	48%
2	104	495	599	18	36	15	69	88%
3	98	737	835	19	51	15	85	89%
4	114	1734	1848	34	86	15	135	92%

Tests were run on an Intel core i9 computer, with 8 cores at 2.3GHz, 32 Go RAM, running MacOS and Java 8, with TTool build version 13854 (August 2021).

Application to Model Refinement

Avoiding redoing proofs not impacted by a model refinement



Conclusion

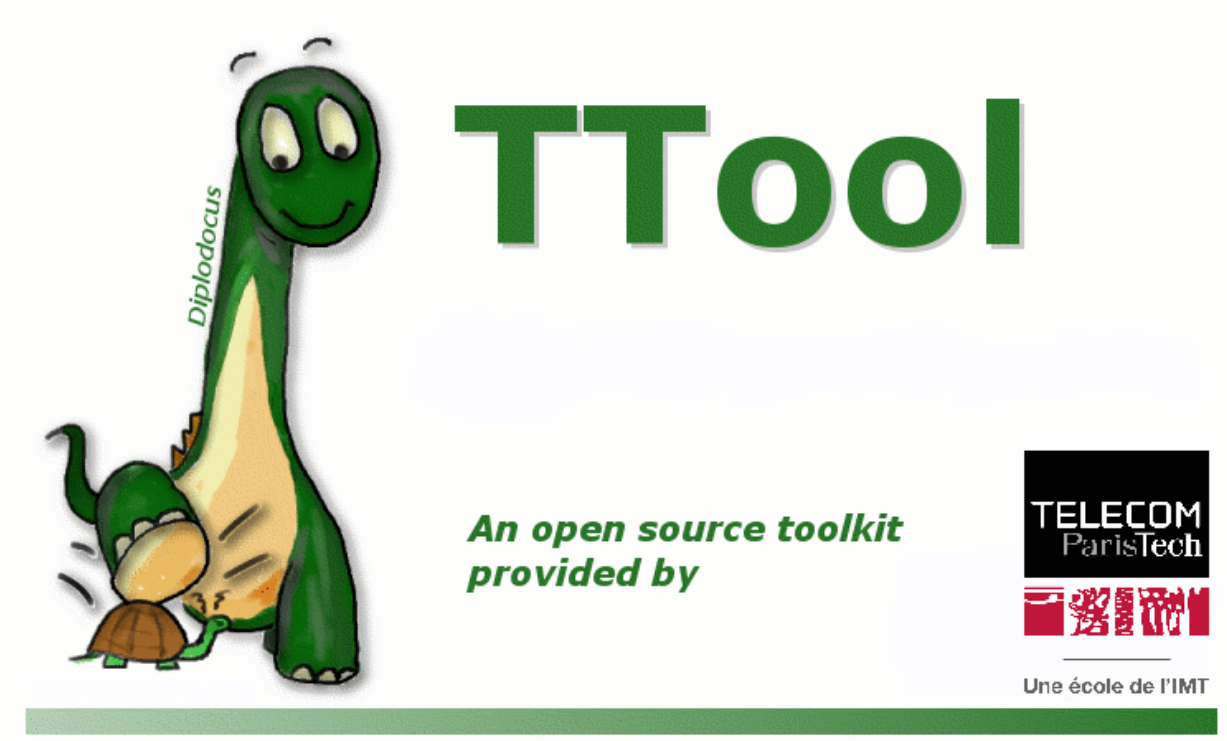
Improving model-checking

- Model adaptation for each property
- Use of dependency graphs
- Important gain demonstrated on one case study

Future work

- Other case studies
- Model refinement: defining bisimulation relation
- Extending dependencies to SysML allocations

To Go Further...



ttool.telecom-paris.fr