Requirements Formalization and Verification

In the context of adaptive CPSs, checking the consistency of requirements is an indisputable, yet challenging task.

- ❑ Requirements written in natural language call for time-consuming and error-prone manual reviews, BUT
- ❑ enabling automated consistency verification often requires overburdening formalizations.

We need practical solutions to enable automated verification of requirements!

Formal Specification

Goal: (Semi) Automatic Translation from Natural Language Specification to Formal Specification.

Desiderata: Unambiguous language with high expressiveness, that can be automatically translated in some logic and then used for verification/validation.

Expressiveness vs Unambiguity!

Actually...

Property Specification Patterns (PSPs) offer a viable path towards this goal.

- ❑ PSP: collection of parameterizable, high-level, formalism-independent specification abstractions
  - ○ developed to capture recurring solutions to the needs of requirement engineering.
- ❑ Each pattern can be directly encoded in a formal specification language
  - ○ linear time temporal logic (LTL), computational tree logic (CTL) ….

PSPs may ease the burden of formalizing requirements, yet enable their verification using automated reasoning tools (e.g., for LTL).

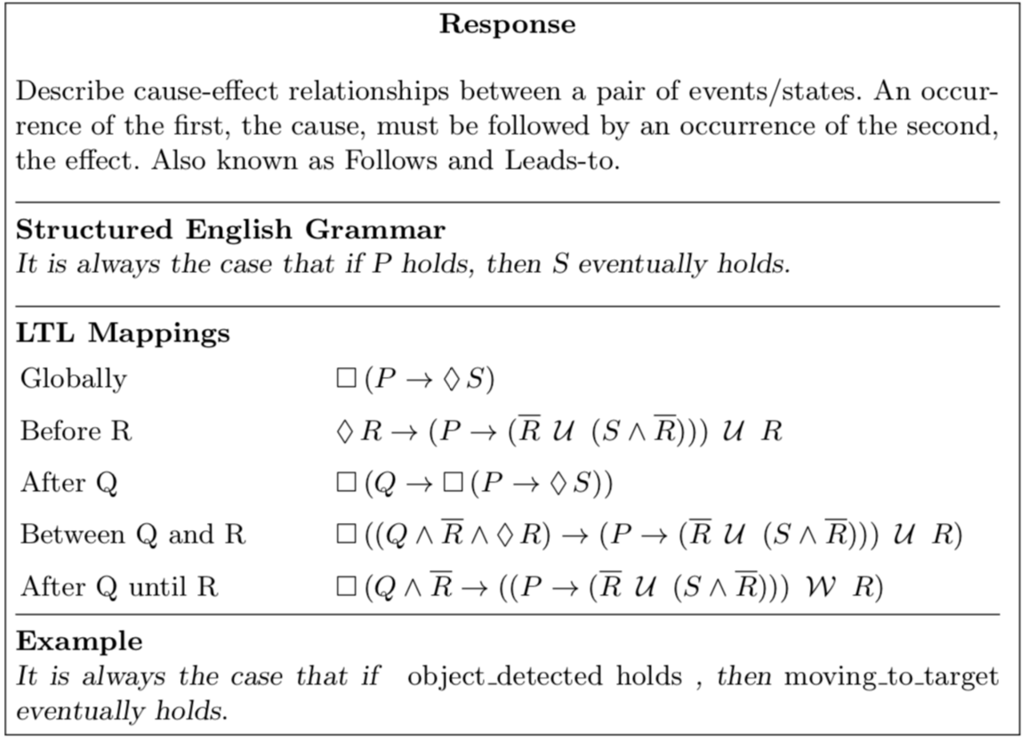
Property Specification Patterns (PSPs)

- ❑ PSPs are meant to describe the essential structure of system's behaviours and provide expressions of such behaviors in a range of common formalisms.
- ❑ A pattern is comprised of a
  - o name;
  - o an informal statement describing the behaviour captured by the pattern;
  - o a (structured English) statement that should be used to express requirements.

Property Specification Patterns (PSPs)

The LTL mappings corresponding to different declinations of the pattern are also given, where capital letters (P, Q, R, ...) stands for Boolean states/events.

A complete list of patterns is available at http://patterns.projects.cs.ksu.edu



Response

Describe cause-effect relationships between a pair of events/states. An occurrence of the first, the cause, must be followed by an occurrence of the second, the effect. Also known as Follows and Leads-to.

Structured English Grammar

It is always the case that if P holds, then S eventually holds.

LTL Mappings

Globally	$\Box(P \rightarrow \Diamond S)$
Before R	$\Diamond R \rightarrow (P \rightarrow (\overline{R}\ \mathcal{U}\ (S \wedge \overline{R})))\ \mathcal{U}\ R$
After Q	$\Box(Q \rightarrow \Box(P \rightarrow \Diamond S))$
Between Q and R	$\Box((Q \wedge \overline{R} \wedge \Diamond R) \rightarrow (P \rightarrow (\overline{R}\ \mathcal{U}\ (S \wedge \overline{R})))\ \mathcal{U}\ R)$
After Q until R	$\Box(Q \wedge \overline{R} \rightarrow ((P \rightarrow (\overline{R}\ \mathcal{U}\ (S \wedge \overline{R})))\ \mathcal{W}\ R)$

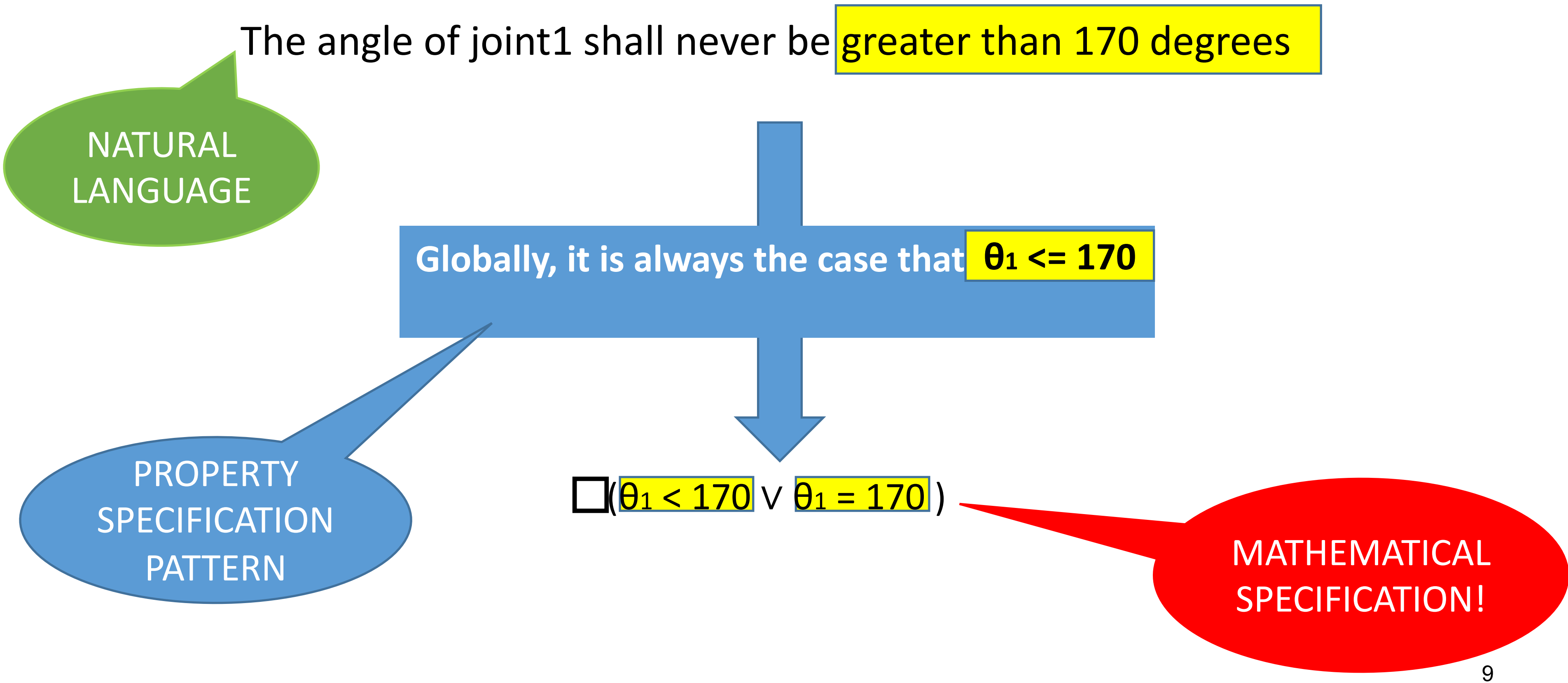
Example

It is always the case that if object_detected holds , *then* moving_to_target *eventually holds.*

Extending PSPs

- The original formulation of PSPs caters for temporal structure over Boolean variables: **for most practical applications, such expressiveness is too restricted.**
- Example: embedded controller for robotic manipulators (from CERBERO use case)
  - With original PSPs, requirements such as "*The angle of joint1 shall never be greater than 170 degrees*" cannot be expressed.
- Solution proposed in CERBERO: PSPs with Boolean and Constrained Numerical Signals (with sound translation to LTL).

Summing up…



The angle of joint1 shall never be greater than 170 degrees

NATURAL LANGUAGE

Globally, it is always the case that $\theta_1 <= 170$

PROPERTY SPECIFICATION PATTERN

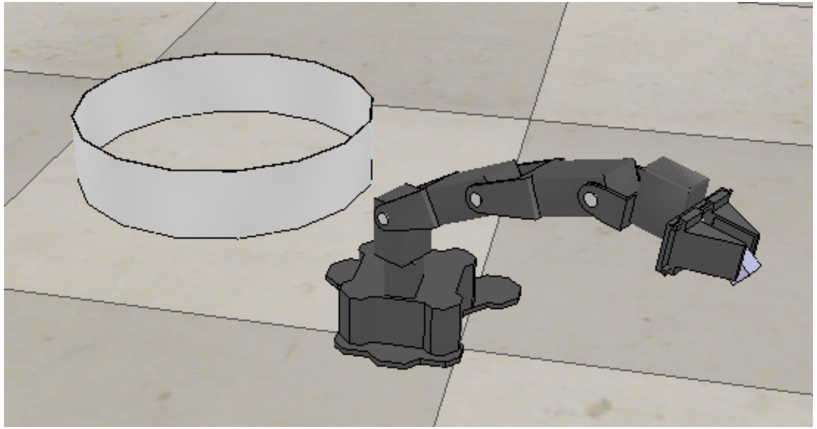
$\Box(\theta_1 < 170 \lor \theta_1 = 170)$

MATHEMATICAL SPECIFICATION!

Controller for a Robotic Manipulator

Let consider a set of requirements from the design of an embedded controller for a robotic manipulator:

- The controller should direct a properly initialized robotic arm to look for an object placed in a given position and move to such position in order to grab the object.
- Once grabbed, the object is to be moved into a bucket placed in a given position and released without touching the bucket.
- The robot must stop also in the case of an unintended collision with other objects or with the robot itself.
- Collisions can be detected using torque estimation from sensors placed in the joints.



The manipulator is a 4 degrees-of-freedom Trossen Robotics WidowX Arm equipped with a gripper

Requirements

- ❑ Constrained numerical signals are used to represent requirements related to various parameters
  - o angle, speed, acceleration, and torque of the 4 joints, size of the object picked, and force exerted by the end-effector.
- ❑ 75 requirements in total.

```
Globally, it is never the case that joint1_angle < -170 or joint1_angle > 170 holds.
                                        …
Globally, it is always the case that if ef_idle holds, then ef_speed = 0 and ef_acc = 0 holds
as well.
                                        …
After state_init until state_scanning, it is never the case that state_moving_to_target
holds.
```

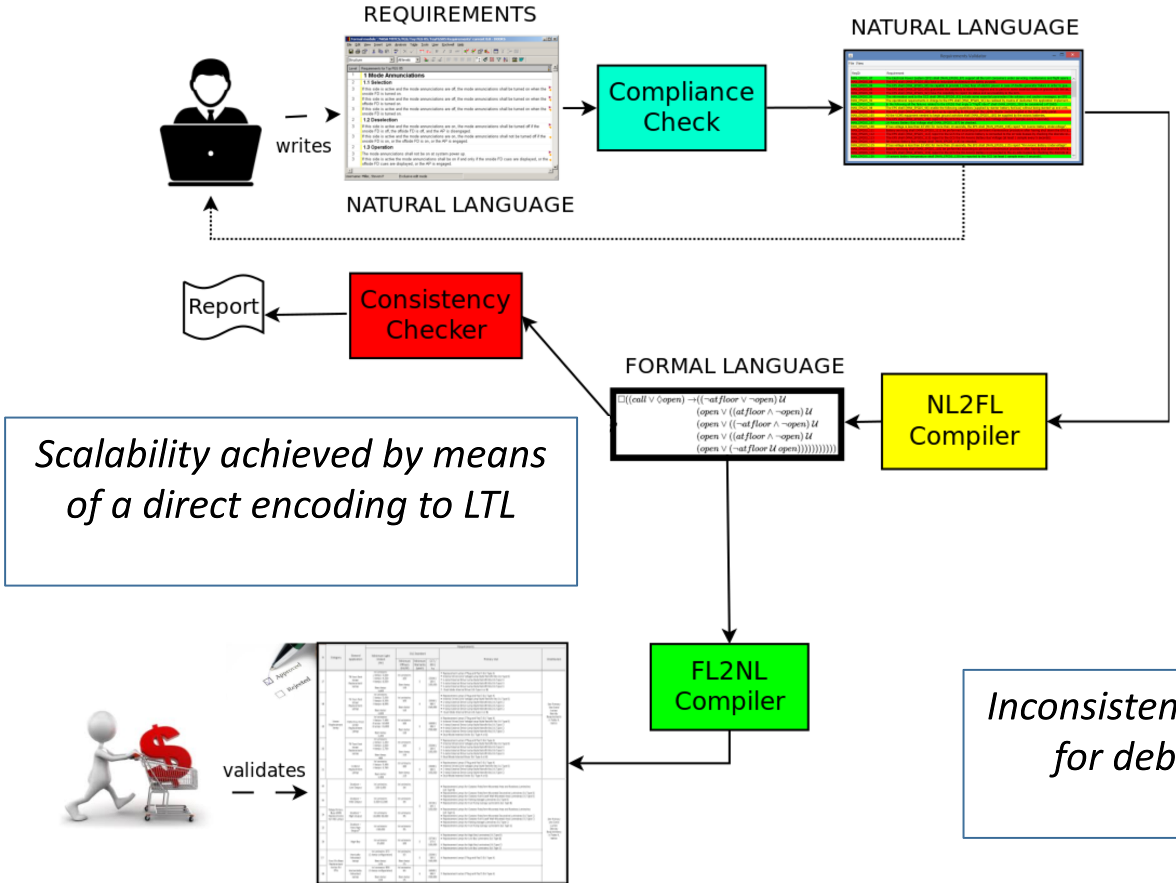
The complete list is available at https://github.com/SAGE-Lab/robot-arm-usecase

See the use case document.

Consistency checking

- The formal representation of all requirements is "glued" together.
- The resulting formula is checked with a Model Checker.
- If the formula is satisfiable, then the system can be realized.
- Otherwise, **inconsistency => Impossible to build a system that satisfy all the requirements!**

The ReqV tool



$$\Box((call \vee \Diamond open) \rightarrow ((\neg at\,floor \vee \neg open)\,\mathcal{U}\,(open \vee ((at\,floor \wedge \neg open)\,\mathcal{U}\,(open \vee ((\neg at\,floor \wedge \neg open)\,\mathcal{U}\,(open \vee ((at\,floor \wedge \neg open)\,\mathcal{U}\,(open \vee (\neg at\,floor\,\mathcal{U}\,open))))))))))$$

Scalability achieved by means of a direct encoding to LTL

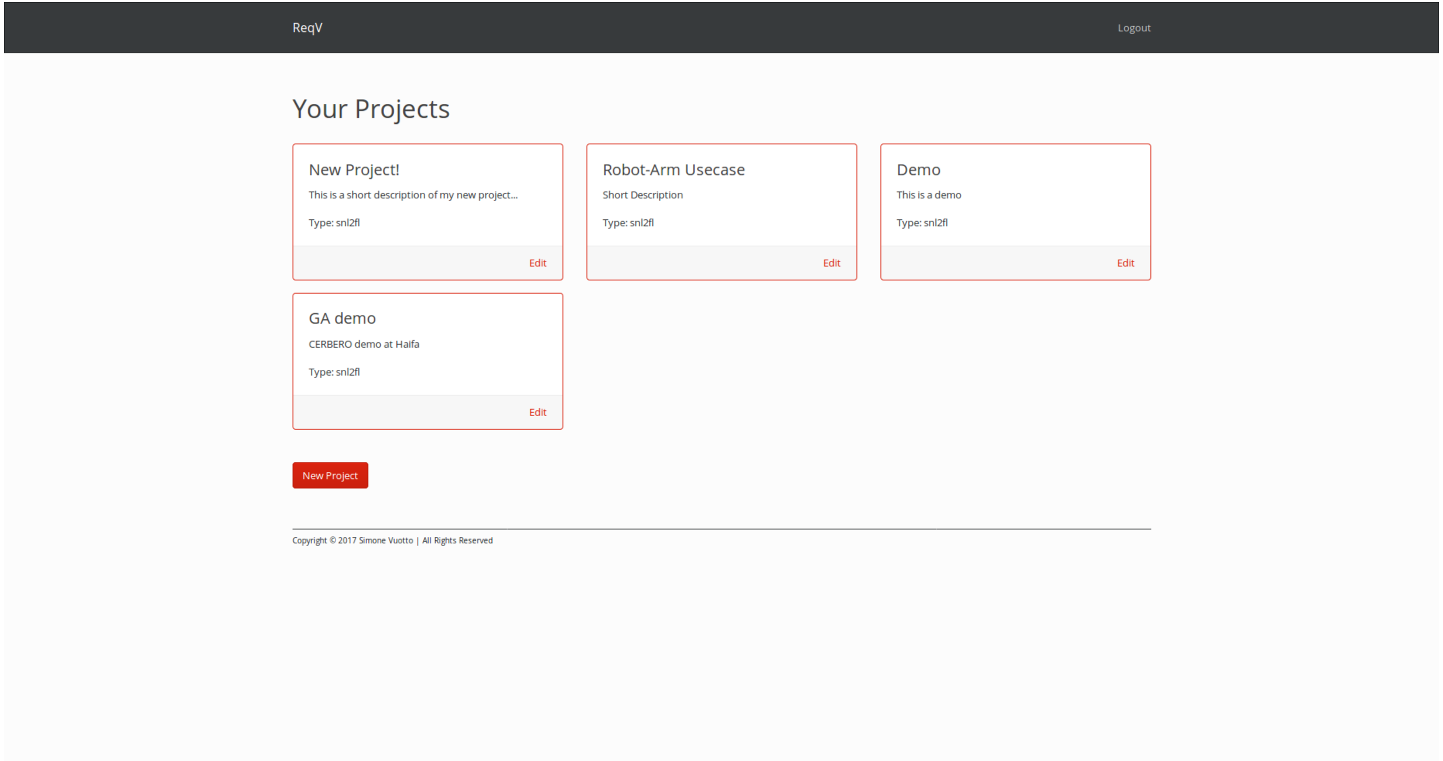
Joint work with Armando Tacchella, Massimo Narizzano, and Simone Vuotto

Avaliable at
https://gitlab.sagelab.it/sage/ReqV

Evaluate ReqV at
https://reqv.sagelab.it/

Inconsistency finder procedure for debugging purpose

The ReqV tool



Your Projects

New Project!

This is a short description of my new project...

Type: snl2fl

Robot-Arm Usecase

Short Description

Type: snl2fl

Demo

This is a demo

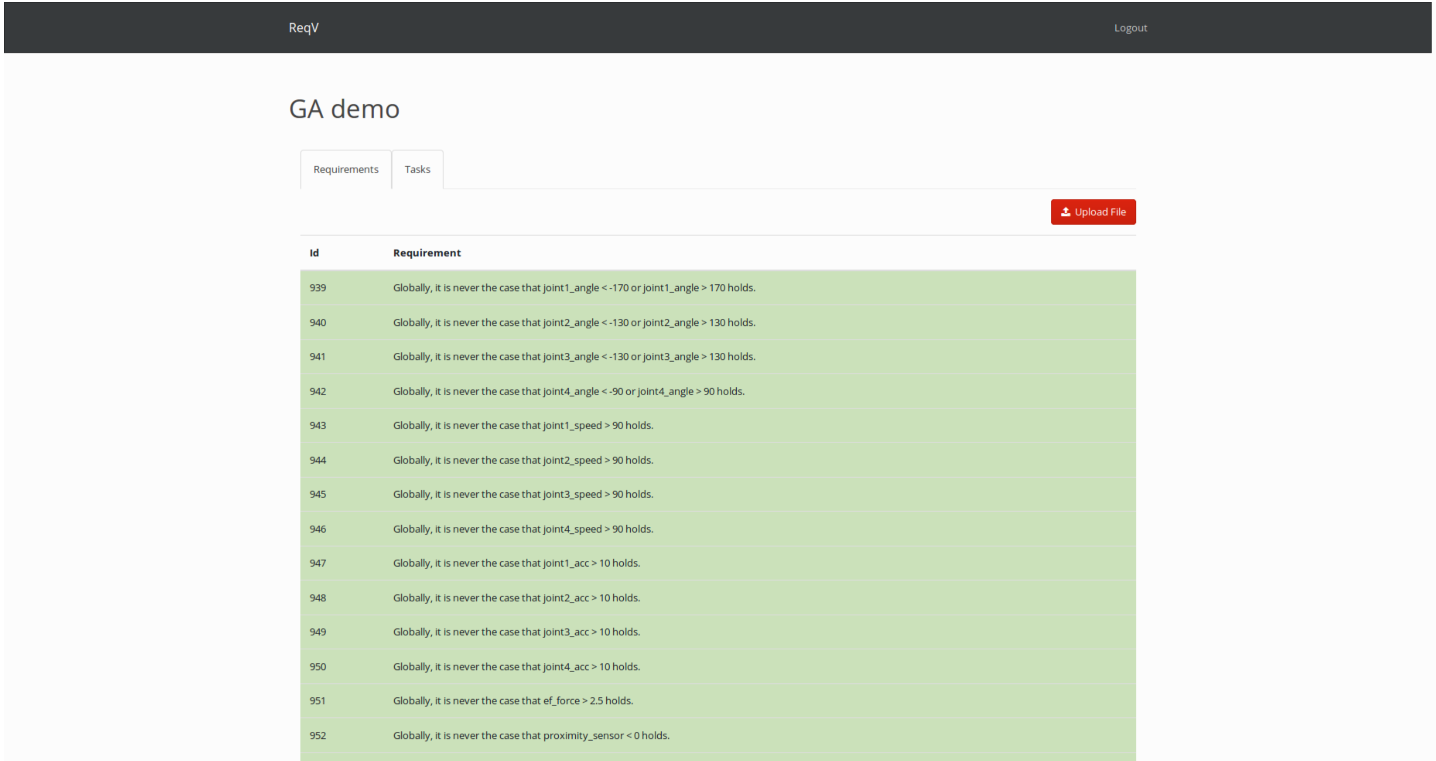
Type: snl2fl

GA demo

CERBERO demo at Haifa

Type: snl2fl

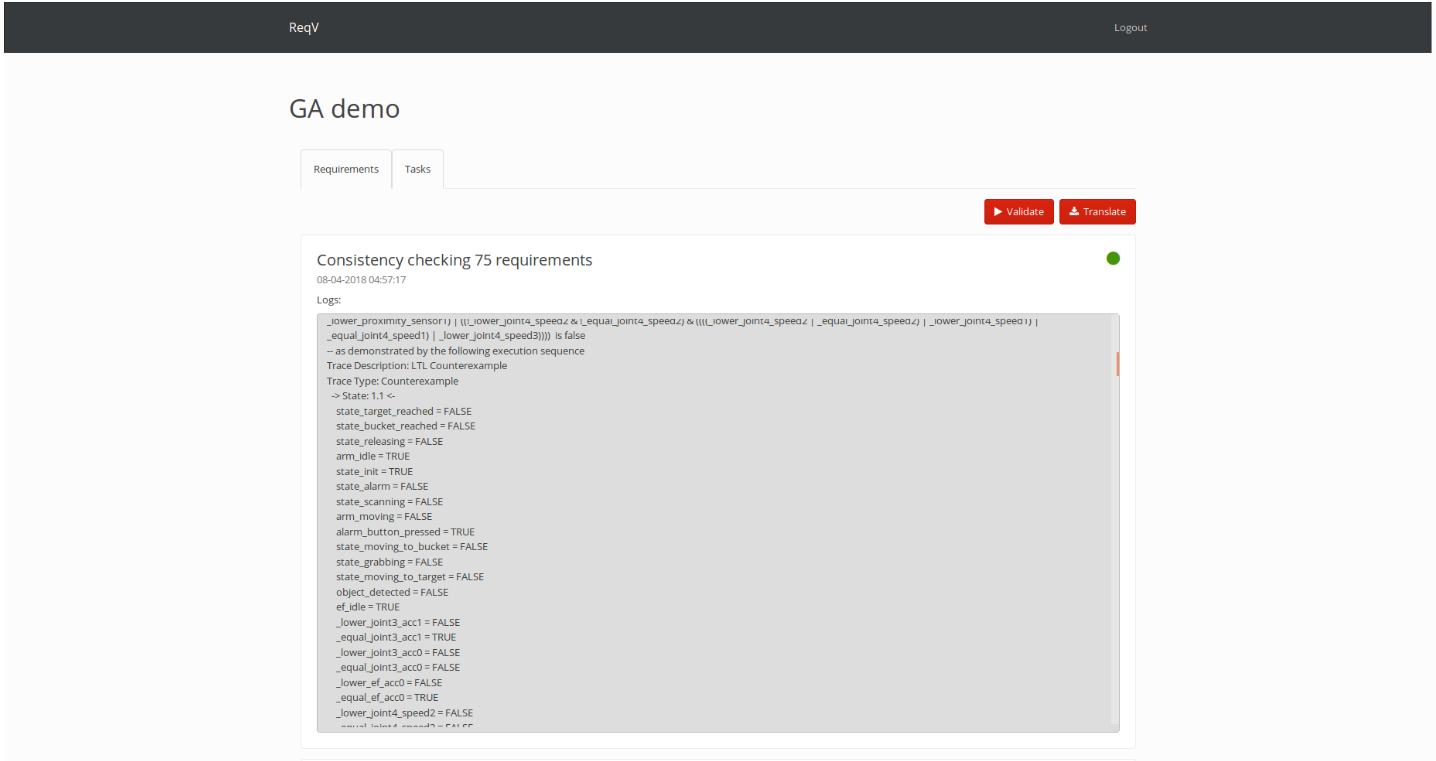
The ReqV tool



GA demo

Id	Requirement
939	Globally, it is never the case that joint1_angle < -170 or joint1_angle > 170 holds.
940	Globally, it is never the case that joint2_angle < -130 or joint2_angle > 130 holds.
941	Globally, it is never the case that joint3_angle < -130 or joint3_angle > 130 holds.
942	Globally, it is never the case that joint4_angle < -90 or joint4_angle > 90 holds.
943	Globally, it is never the case that joint1_speed > 90 holds.
944	Globally, it is never the case that joint2_speed > 90 holds.
945	Globally, it is never the case that joint3_speed > 90 holds.
946	Globally, it is never the case that joint4_speed > 90 holds.
947	Globally, it is never the case that joint1_acc > 10 holds.
948	Globally, it is never the case that joint2_acc > 10 holds.
949	Globally, it is never the case that joint3_acc > 10 holds.
950	Globally, it is never the case that joint4_acc > 10 holds.
951	Globally, it is never the case that ef_force > 2.5 holds.
952	Globally, it is never the case that proximity_sensor < 0 holds.

The ReqV tool



GA demo

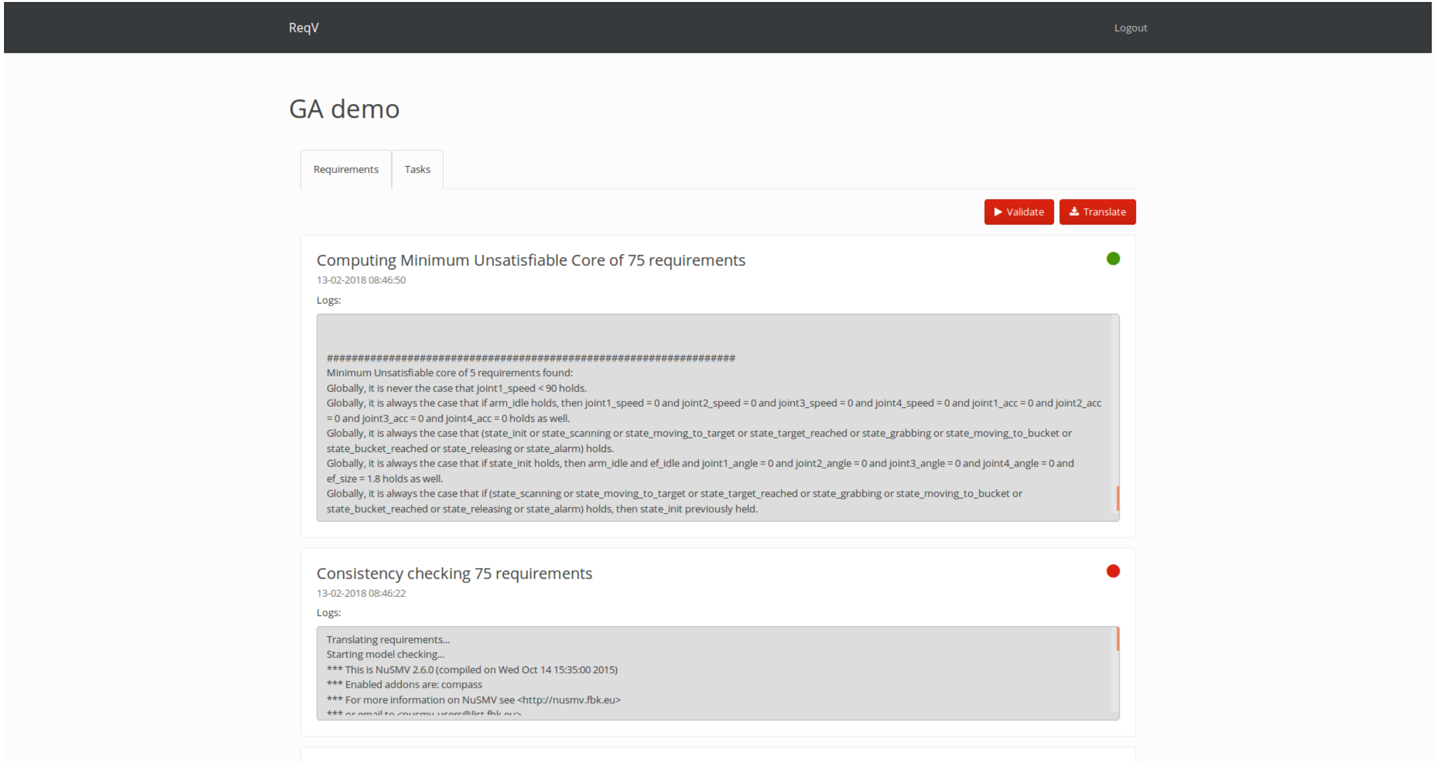
Consistency checking 75 requirements ●

08-04-2018 04:57:17

Logs:

```
_lower_proximity_sensor1) | ((!_lower_joint4_speed2 & !_equal_joint4_speed2) & ((((_lower_joint4_speed2 | _equal_joint4_speed2) | _lower_joint4_speed1) |
_equal_joint4_speed1) | _lower_joint4_speed3)))) is false
-- as demonstrated by the following execution sequence
Trace Description: LTL Counterexample
Trace Type: Counterexample
  -> State: 1.1 <-
    state_target_reached = FALSE
    state_bucket_reached = FALSE
    state_releasing = FALSE
    arm_idle = TRUE
    state_init = TRUE
    state_alarm = FALSE
    state_scanning = FALSE
    arm_moving = FALSE
    alarm_button_pressed = TRUE
    state_moving_to_bucket = FALSE
    state_grabbing = FALSE
    state_moving_to_target = FALSE
    object_detected = FALSE
    ef_idle = TRUE
    _lower_joint3_acc1 = FALSE
    _equal_joint3_acc1 = TRUE
    _lower_joint3_acc0 = FALSE
    _equal_joint3_acc0 = FALSE
    _lower_ef_acc0 = FALSE
    _equal_ef_acc0 = TRUE
    _lower_joint4_speed2 = FALSE
    _equal_joint4_speed2 = FALSE
```

The ReqV tool



GA demo

Computing Minimum Unsatisfiable Core of 75 requirements

13-02-2018 08:46:50

Logs:

```
######################################################################
Minimum Unsatisfiable core of 5 requirements found:
Globally, it is never the case that joint1_speed < 90 holds.
Globally, it is always the case that if arm_idle holds, then joint1_speed = 0 and joint2_speed = 0 and joint3_speed = 0 and joint4_speed = 0 and joint1_acc = 0 and joint2_acc
= 0 and joint3_acc = 0 and joint4_acc = 0 holds as well.
Globally, it is always the case that (state_init or state_scanning or state_moving_to_target or state_target_reached or state_grabbing or state_moving_to_bucket or
state_bucket_reached or state_releasing or state_alarm) holds.
Globally, it is always the case that if state_init holds, then arm_idle and ef_idle and joint1_angle = 0 and joint2_angle = 0 and joint3_angle = 0 and joint4_angle = 0 and
ef_size = 1.8 holds as well.
Globally, it is always the case that if (state_scanning or state_moving_to_target or state_target_reached or state_grabbing or state_moving_to_bucket or
state_bucket_reached or state_releasing or state_alarm) holds, then state_init previously held.
```

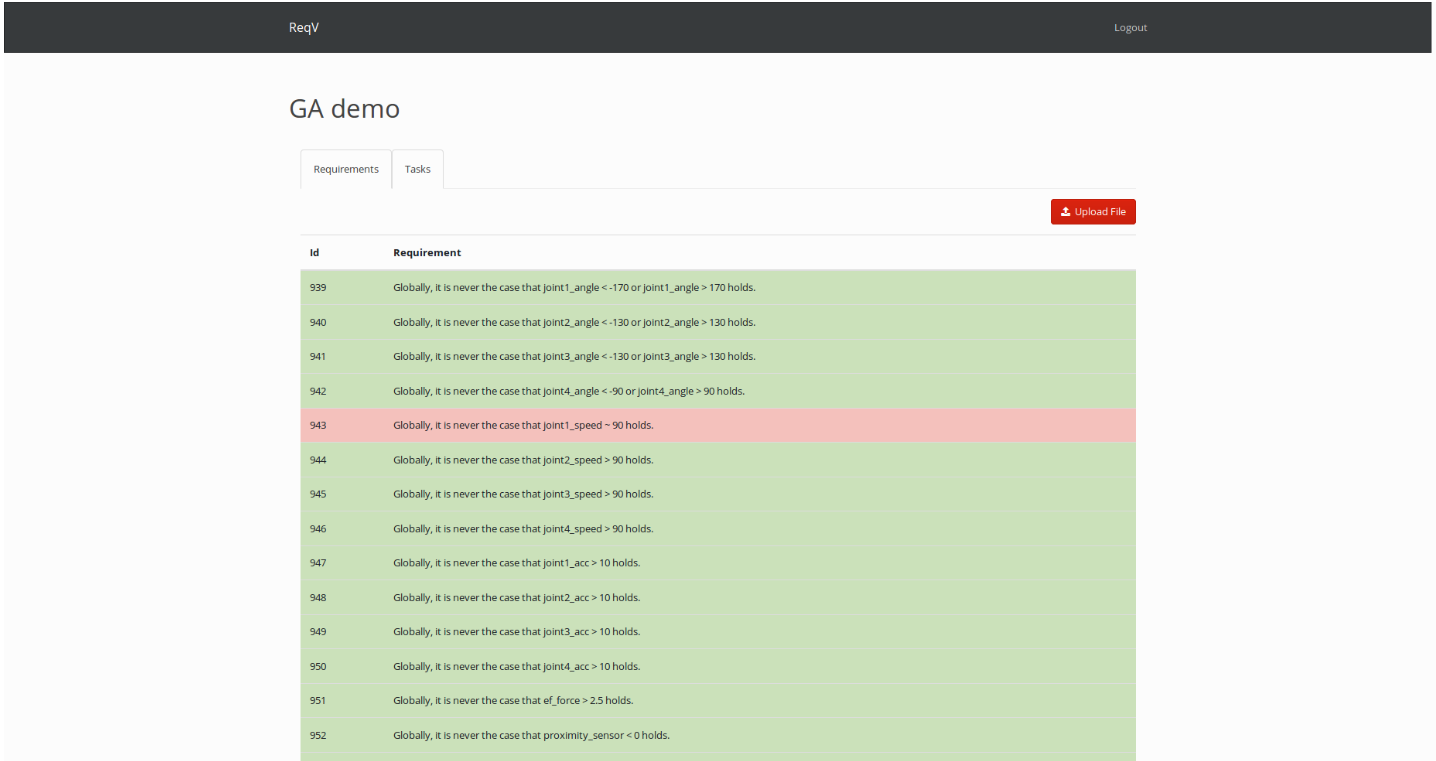
Consistency checking 75 requirements

13-02-2018 08:46:22

Logs:

```
Translating requirements...
Starting model checking...
*** This is NuSMV 2.6.0 (compiled on Wed Oct 14 15:35:00 2015)
*** Enabled addons are: compass
*** For more information on NuSMV see <http://nusmv.fbk.eu>
*** or email to <nusmv-users@list.fbk.eu>
```

The ReqV tool



GA demo

Id	Requirement
939	Globally, it is never the case that joint1_angle < -170 or joint1_angle > 170 holds.
940	Globally, it is never the case that joint2_angle < -130 or joint2_angle > 130 holds.
941	Globally, it is never the case that joint3_angle < -130 or joint3_angle > 130 holds.
942	Globally, it is never the case that joint4_angle < -90 or joint4_angle > 90 holds.
943	Globally, it is never the case that joint1_speed ~ 90 holds.
944	Globally, it is never the case that joint2_speed > 90 holds.
945	Globally, it is never the case that joint3_speed > 90 holds.
946	Globally, it is never the case that joint4_speed > 90 holds.
947	Globally, it is never the case that joint1_acc > 10 holds.
948	Globally, it is never the case that joint2_acc > 10 holds.
949	Globally, it is never the case that joint3_acc > 10 holds.
950	Globally, it is never the case that joint4_acc > 10 holds.
951	Globally, it is never the case that ef_force > 2.5 holds.
952	Globally, it is never the case that proximity_sensor < 0 holds.

Conclusions

- Enabling the automated (formal) verification of requirements is one of the key aspects towards the development of safety- and security-critical CPSs.
- The expressiveness of original PSPs is often too restricted for practical applications.
  - Hybrid systems? Probabilistic models? Real-time constraints?
- Main issue: scalability!

Some references

- PSPs: Dwyer, Matthew B., George S. Avrunin, and James C. Corbett. "Patterns in property specifications for finite-state verification." *Proceedings of the 21st international conference on Software engineering*. ACM, 1999.
- LTL: Pnueli, Amir, and Zohar Manna. "The temporal logic of reactive and concurrent systems." *Springer* 16 (1992): 12.
- NuSMV model checker: Cimatti, A., Clarke, E., Giunchiglia, E., Giunchiglia, F., Pistore, M., Roveri, M., ... & Tacchella, A. (2002, July). Nusmv 2: An opensource tool for symbolic model checking. In *International Conference on Computer Aided Verification* (pp. 359-364). Springer.
- Model checking: Baier, Christel, Joost-Pieter Katoen, and Kim Guldstrand Larsen. *Principles of model checking*. MIT press, 2008.
- PSPs with boolean and constrained numerical signals: Narizzano, M., Pulina, L., Tacchella, A., & Vuotto, S. (2018, April). Consistency of Property Specification Patterns with Boolean and Constrained Numerical Signals. In *NASA Formal Methods Symposium* (pp. 383-398). Springer, Cham.