Analyzing and Debugging Normative Requirements via Satisfiability Checking

Nick Feng, **Lina Marsso**,
Sinem Getir Yaman, Bev Townsend, Yesugen Baatartogtokh, Reem Ayad, Ioannis Stefanakos,
Victória Oldemburgo de Mello, Isobel Standen, Calum Imrie, Genaina Rodrigues,
Ana Cavalcanti, Radu Calinescu, and Marsha Chechik

ICSE 2024
April 18, 2024

UNIVERSITY OF TORONTO
UNIVERSITY of York
Universidade de Brasília

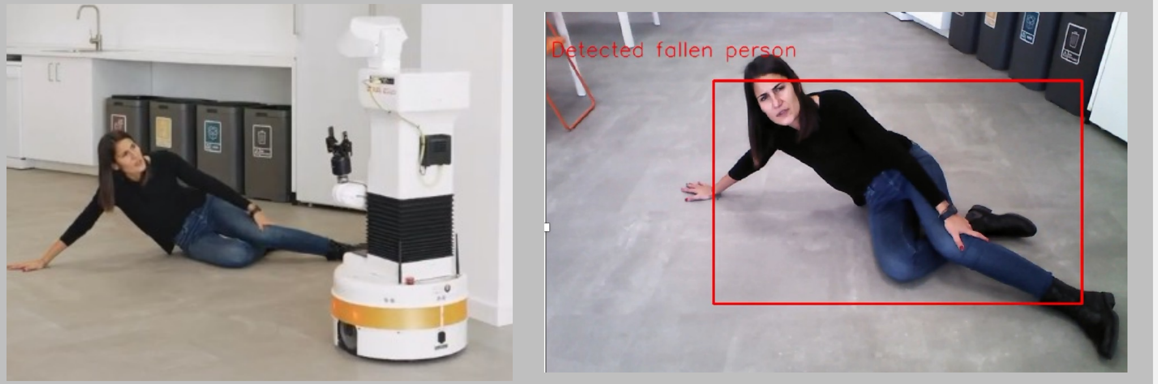
Context

Systems increasingly interacting with humans in various domains

(transport, environment, health and social care)

ALMI: Assistive-care robotics

Helps with food preparation, dressing, fallen-user alert, etc.



Detect the user has fallen

Alert that the user has fallen

ALMI robot from RoboStar (University of York, UK)

Normative Requirements

- Capture **s**ocial, **l**egal, **e**thical, **e**mpathetic, **c**ultural (**SLEEC**) aspects of systems

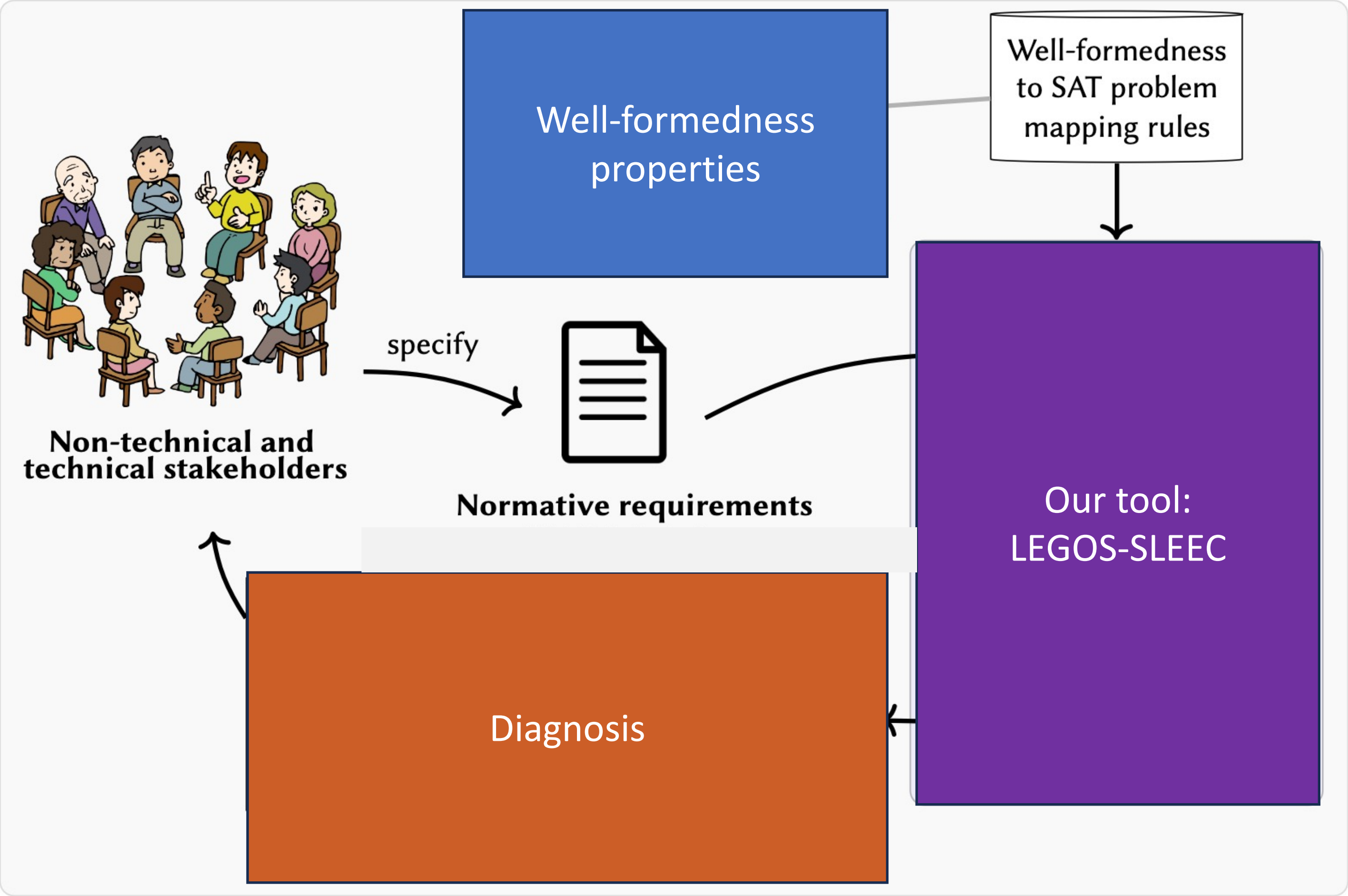
- Specified by stakeholders with non-technical expertise
  - Lawyer, regulators, ethicists, etc.

- Hard to get right
  - Stakeholders from different fields, different vocabularies
  - Their views are often conflicting or redundant
  - Stakeholders might not have sufficient technical background to reason about requirements
  - Requirements are complex: Allow constraints over data and time

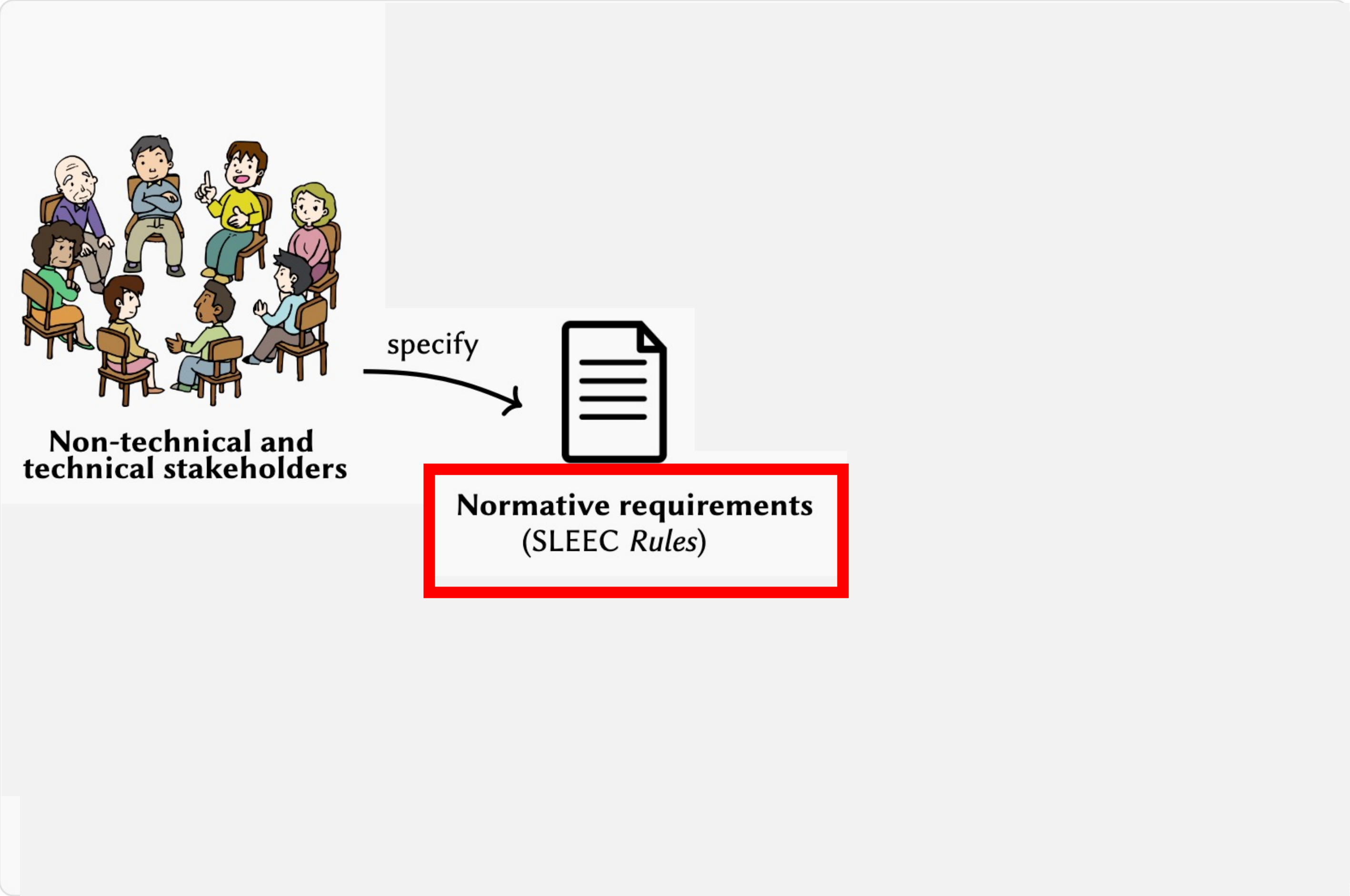
Our goal

Helping non-technical stakeholders elicit a coherent and well-formed set of normative requirements

Overview of the proposed approach



Outline



I. Background: SLEEC DSL

Background: SLEEC DSL [GYBJCC23]

Definitions

definition block

event DressingStarted
event CurtainOpenRqst
event CurtainsOpened

measure userUnderdressed: **Bool**
measure roomTemperature: **numeric**

Events: atomic, instantaneous actions

Measures: readable properties, types: *Boolean, numeric*

Rules

rule block

Rule1 **when** CurtainOpenRequest **then** CurtainsOpened **within** 30 **seconds**

unless userUnderDressed **then** RefuseRequest **within** 30 **seconds**

[GYBJCC23] S. Getir-Yaman, C. Burholt, M. Jones, R. Calinescu, and A. Cavalcanti. "Specification and Validation of Normative Rules for Autonomous Agents", FASE 2023.

Background: SLEEC DSL [GYBJCC23]

definition block

Definitions

event DressingStarted
event CurtainOpenRqst
event CurtainsOpened

Events: atomic, instantaneous actions

measure userUnderdressed: **Bool**
measure roomTemperature: **numeric**

Measures: readable properties, types: *Boolean, numeric*

rule block

Rules

Base rule

Rule1 **when** CurtainOpenRequest **then** CurtainsOpened **within** 30 **seconds**

unless userUnderDressed **then** RefuseRequest **within** 30 **seconds**

Defeater

[GYBJCC23] S. Getir-Yaman, C. Burholt, M. Jones, R. Calinescu, and A. Cavalcanti. "Specification and Validation of Normative Rules for Autonomous Agents", FASE 2023.

Background: SLEEC DSL [GYBJCC23]

Definitions

definition block

event DressingStarted
event CurtainOpenRqst
event CurtainsOpened
event CallSuport
measure userUnderdressed: **Bool**
measure roomTemperature: **numeric**

Events: atomic, instantaneous actions

Measures: readable properties, types: *Boolean, numeric*

Rules

rule block

TRIGGER: logical expression combining event & measures

Response

Deadline

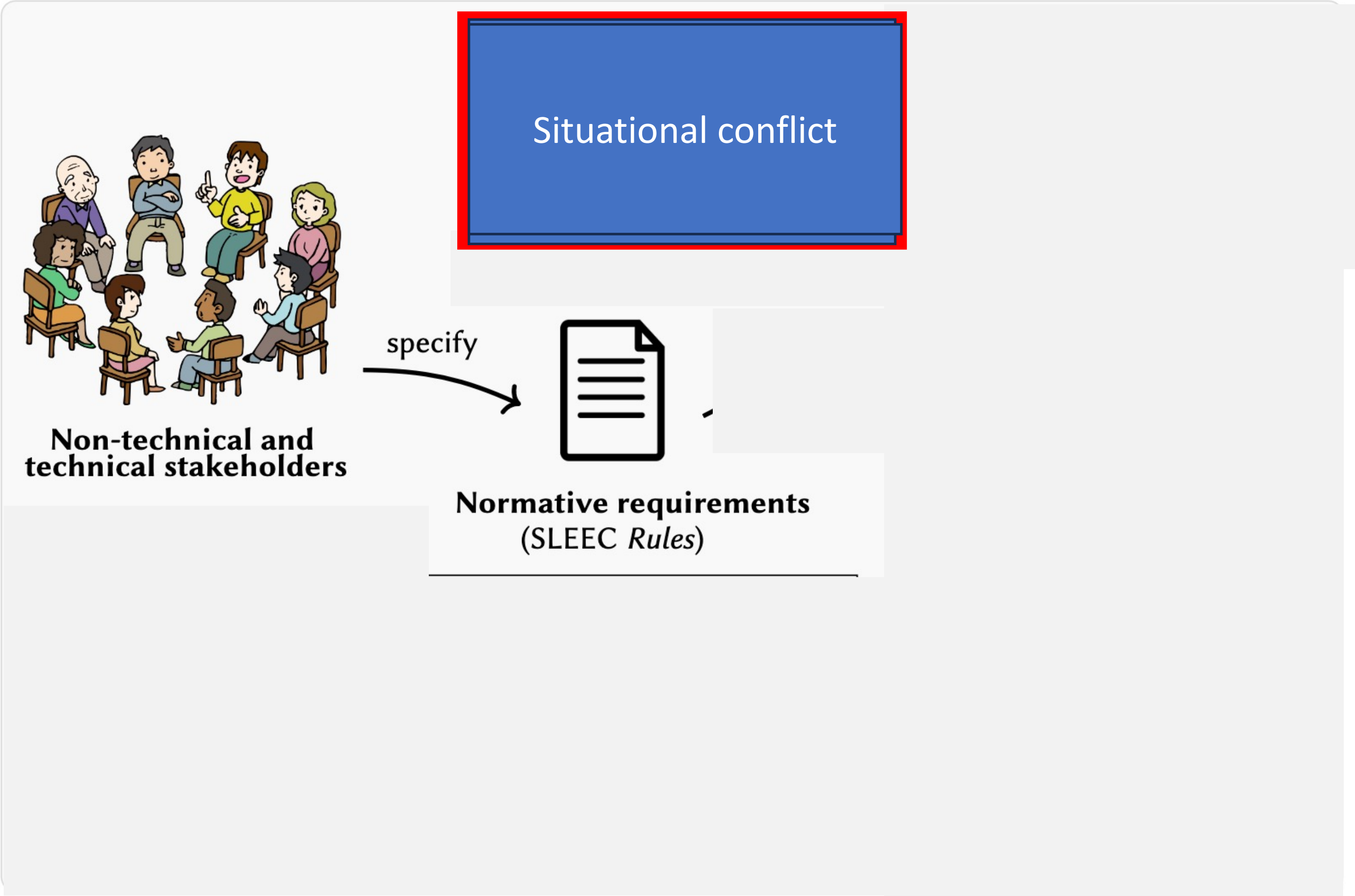
Rule1 **when** CurtainOpenRequest **then** CurtainsOpened **within** 30 **seconds**

otherwise CallSuport

Fallback

[GYBJCC23] S. Getir-Yaman, C. Burholt, M. Jones, R. Calinescu, and A. Cavalcanti. "Specification and Validation of Normative Rules for Autonomous Agents", FASE 2023.

Outline



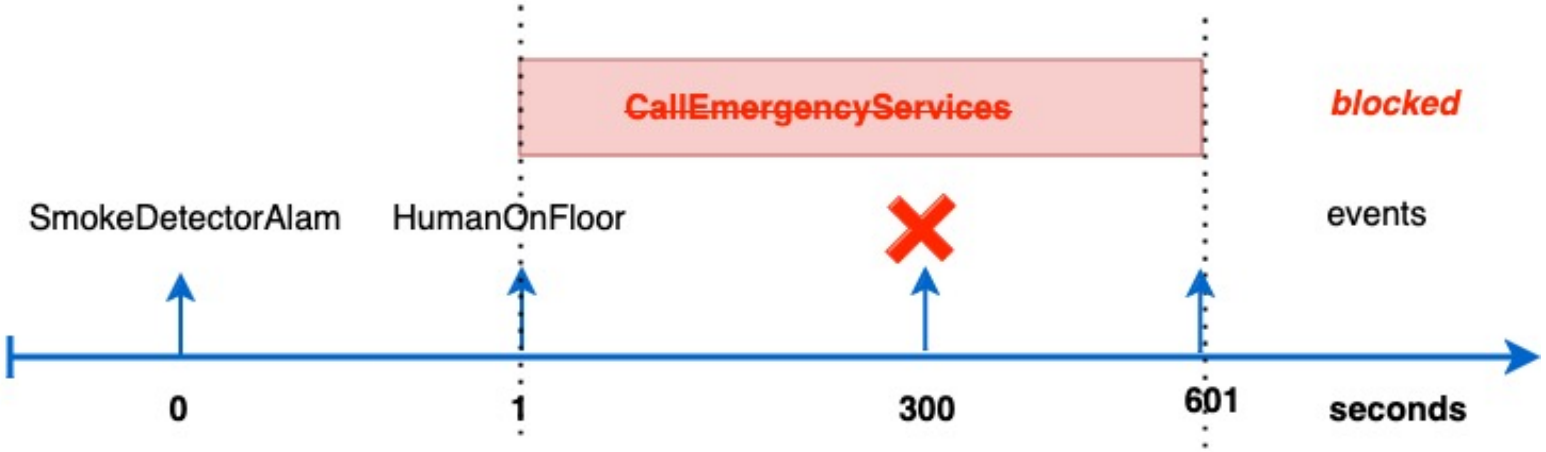
II. Well-formedness properties

Situational conflicts

A given requirement is situationally conflicting if there exists a feasible situation that eventually causes a conflict.

Example:

Situational conflict under situation:



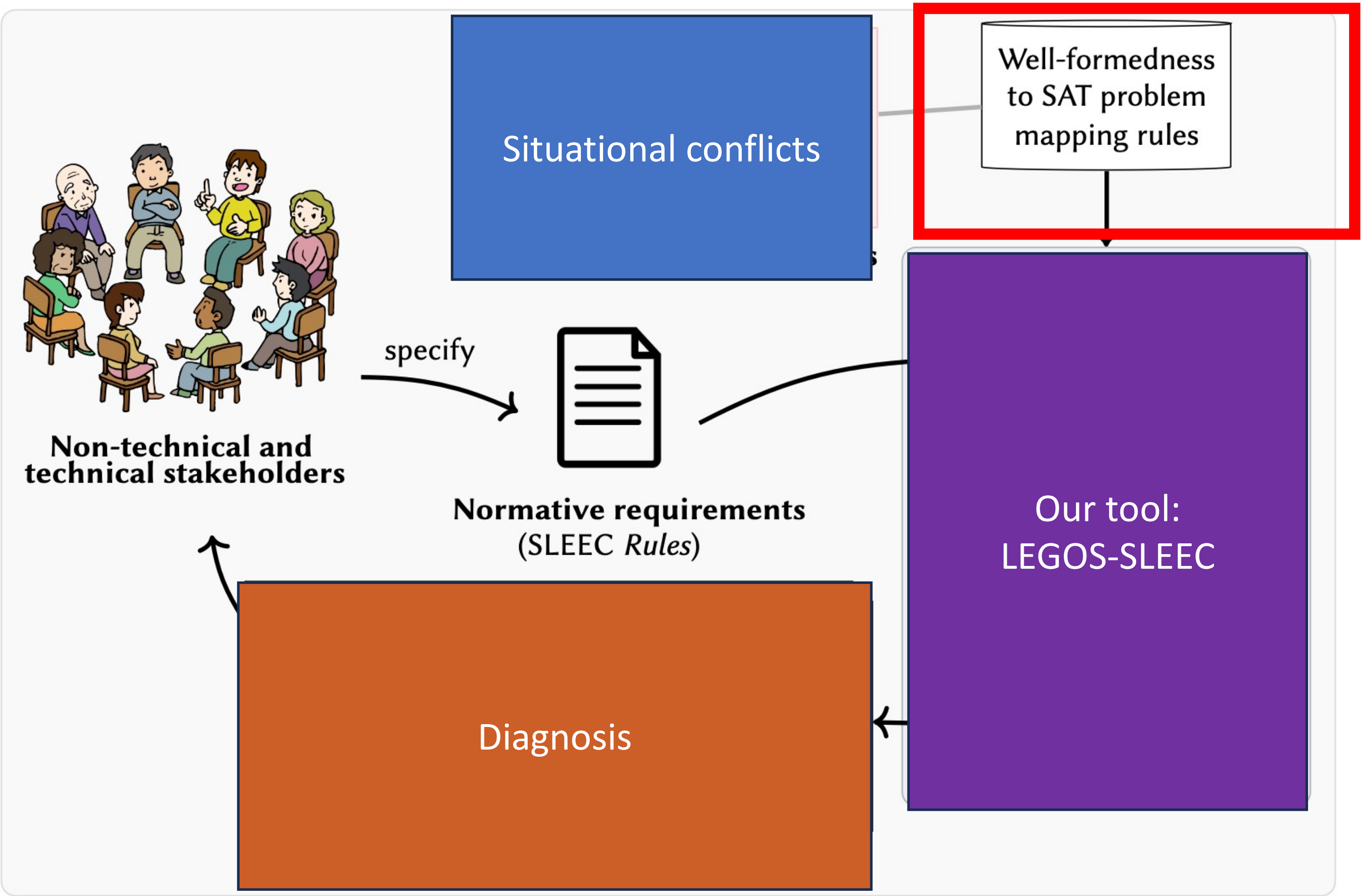
For rule:

R3 **when** HumanOnFloor **and (not** humanAssents) **then not** CallEmergencyServices **within** 600 **seconds**

Because of the following rule:

R21 **when** SmokeDetectorAlarm **then** CallEmergencyServices **within** 300 **seconds**

Outline



III. Well-formedness and satisfiability

Situational conflict verification via satisfiability

1. To find a situation, use backward reasoning symbolically : does there exists a sufficient condition, situation s, such that a rule *ri* is triggered but with the **responses blocked**

Rule1 **when** A **then** B **within** 30 **seconds otherwise** C **within** 5 **seconds**

Situational conflict verification via satisfiability

1. To find a situation, use backward reasoning symbolically : does there exists a sufficient condition, situation s, such that a rule *ri* is triggered but with the **responses blocked**

Rule1 **when** A **then** B **within** 30 **seconds** **otherwise** C **within** 5 **seconds**

We want to block the two responses

Situational conflict verification via satisfiability

1. To find a situation, use backward reasoning symbolically : does there exists a sufficient condition, situation s, such that a rule *ri* is triggered but with the **responses blocked**

Rule1 **when** A **then** B **within** 30 **seconds** **otherwise** C **within** 5 **seconds**

Looking for some rules with a response not B ← Block B within 30 seconds

We want to find some rules that block the response

Situational conflict verification via satisfiability

1. To find a situation, use backward reasoning symbolically : does there exists a sufficient condition, situation s, such that a rule *ri* is triggered but with the **responses blocked**

Rule1 **when** A **then** B **within** 30 **seconds** **otherwise** C **within** 5 **seconds**

Looking for some rules with a response not B ← Block B within 30 seconds

Rule2 **when** D **then** C **within** 30 **seconds** **otherwise not B within** 40 **seconds**

We want to force the necessary condition to block the response

Situational conflict verification via satisfiability

1. To find a situation, use backward reasoning symbolically : does there exists a sufficient condition, situation s, such that a rule *ri* is triggered but with the **responses blocked**

Rule1 **when** A **then** B **within** 30 **seconds** **otherwise** C **within** 5 **seconds**

Looking for some rules with a response not B ← Block B within 30 seconds

Rule2 **when** D **then** C **within** 30 **seconds** **otherwise not B within** 40 **seconds**

C **within** 30 **seconds** → Block C within 30 seconds

We want to force the necessary condition to block the response

Situational conflict verification via satisfiability

1. To find a situation, use backward reasoning symbolically : does there exists a sufficient condition, situation s, such that a rule *ri* is triggered but with the **responses blocked**

Rule1 **when** A **then** B **within** 30 **seconds** **otherwise** C **within** 5 **seconds**

Looking for some rules with a response not B ← Block B within 30 seconds

Rule2 **when** D **then** C **within** 30 **seconds** **otherwise not B within** 40 **seconds**

C **within** 30 **seconds** → Block C within 30 seconds

Looking for some rules with a response not C ← Block C within 30 seconds

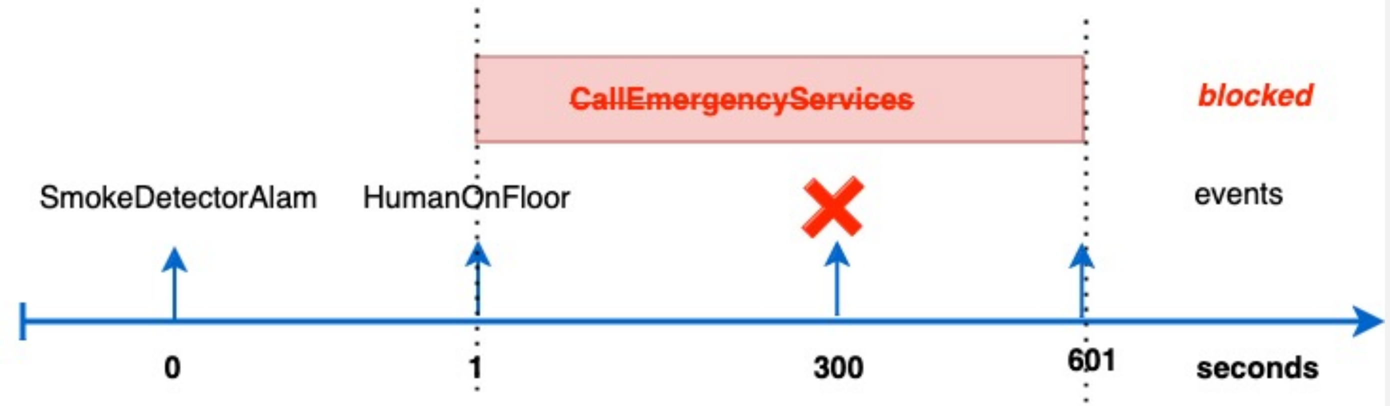
….. (and other constraints)

Backward analysis guarantee to terminate

Situational conflict verification via satisfiability

1. To find a situation, use backward reasoning symbolically: does there exists a sufficient condition, situation s, such that a rule *ri* is triggered but with the response blocked

situation s:



2. To obtain the diagnosis: When such situation exist, we encode s symbolically, and then check whether the entire rule set R and s is UNSAT, we use the UNSAT proof to build a diagnosis

situation s +

For rule:
R3 **when** HumanOnFloor **and (not** humanAssents)
then not CallEmergencyServices **within** 600 **seconds**

Because of the following rule:
R21 **when** SmokeDetectorAlarm **then** CallEmergencyServices **within** 300 **seconds**

Well-formedness issues (WFI)s

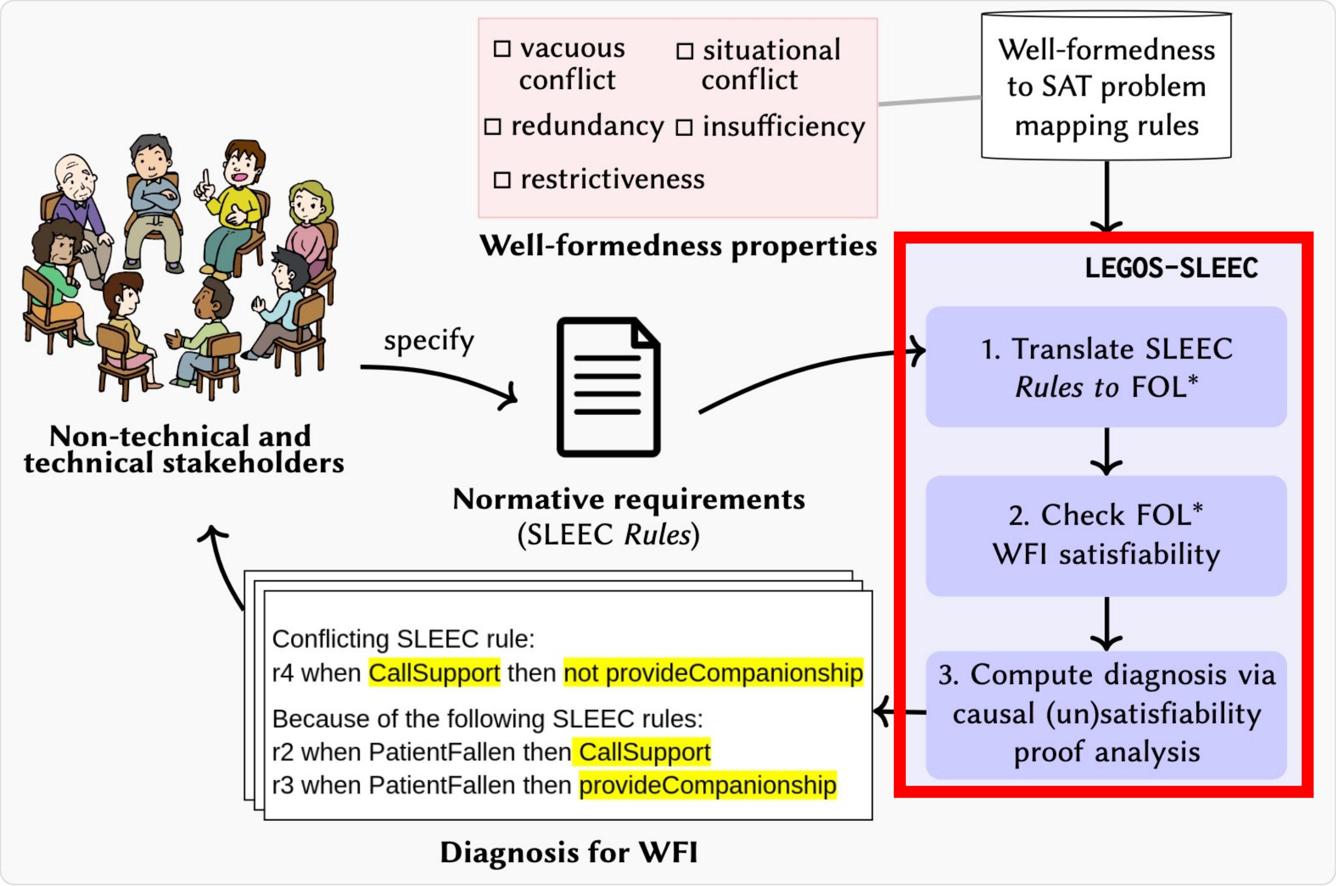
1. Situational conflict and vacuous conflict
2. Restrictive or Insufficient requirements
3. Unecessary redundant requirements

Please find more details in the paper

WFI automatic validation

LEGOS-SLEEC:

- Checks requirements WFIs via FOL* satisfiability checking [CAV23]
- Produces a diagnosis in SLEEC DSL



LEGOS-SLEEC tool: https://github.com/NickF0211/LEGOS-SLEEC

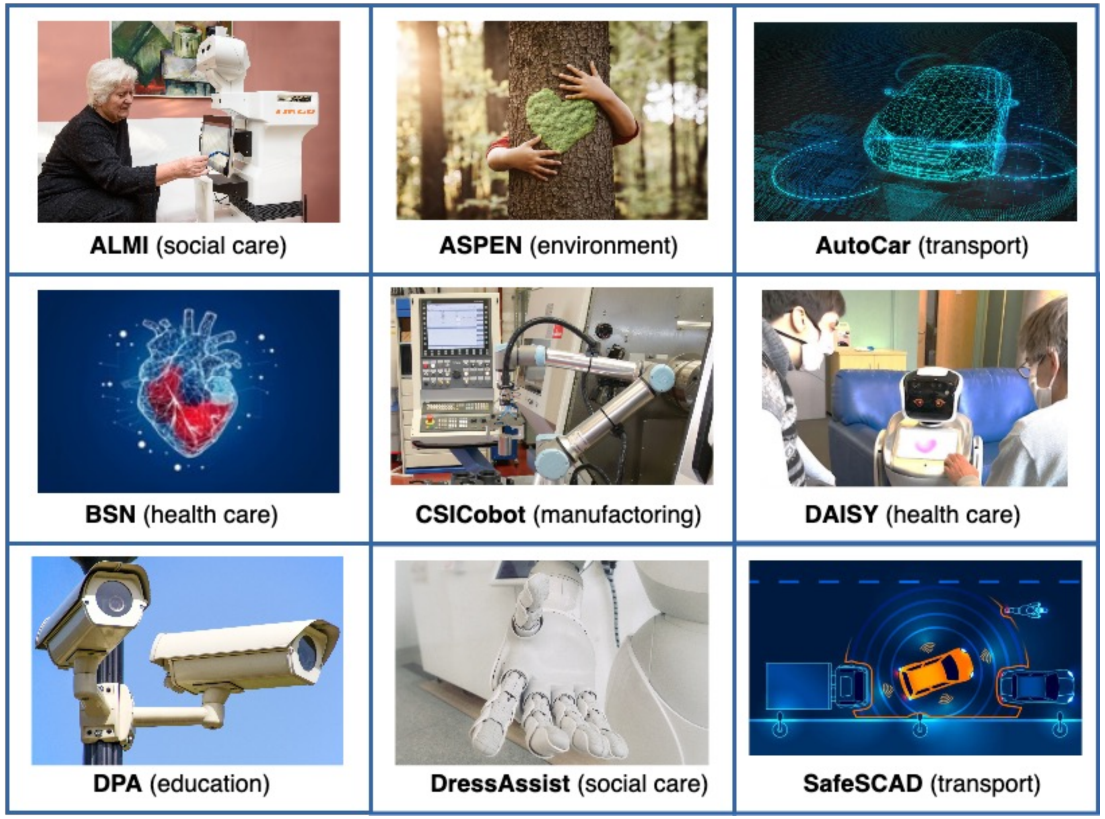
[CAV23] N. Feng, L. Marsso, M. Sabetzadeh, and M. Chechik. "Early verification of legal compliance via bounded satisfiability checking", CAV 2023.

Outline

IV. Evaluation

RESERVE: repository of 9 real-world case studies

- **Domains:** transport, environment, manufacturing health and social care.
- **Different stages:** ranging from the design phase to deployed systems
- **Non-technical stakeholders:** an ethicist, a lawyer, a philosopher, and a psychologist
- **Technical stakeholders:** a safety analyst, and 3 engineers
- **Normative requirements**: 233 N-NFRs in total



RESERVE: http://www.cs.toronto.edu/~sleec/

How effective is LEGOS-SLEEC in detecting WFIs?

For each case study, 1-2 TSs were paired with 1-4 N-TSs:

- Built a set of normative requirements
- Met to manually review, discuss, and agree on these requirements

For each WFI identified by LEGOS-SLEEC, we recorded:

N-TS ability to understand the feedback given by LEGOS-SLEEC
and split the identified WFIs into relevant/spurious

How effective is LEGOS-SLEEC in detecting WFIs?

case studies	v-conf. (TP - FP)	s-conf. (TP - FP)	redund. (TP - FP)	restrict. (TP - FP)	insuffi. (TP - FP)	time (sec.)
ALMI	0 - 0	3 - 0	0 - 0	0 - 0	1 - **1**	30
ASPEN	0 - 0	3 - 0	1 - 0	0 - 0	5 - 0	25.3
AutoCar	0 - 0	4 - 0	2 - 0	0 - 0	9 - 0	27.7
BSN	0 - 0	0 - 0	0 - 0	0 - 0	3 - 0	46
DressAssist	0 - 0	1 - 0	0 - 0	0 - 0	1 - **3**	20.3
CSI-Cobot	0 - 0	0 - 0	2 - 0	0 - 0	6 - **1**	25.3
DAISY	0 - 0	1 - 0	1 - 0	0 - 0	5 - 0	30.4
DPA	0 - 0	0- 0	0 - 0	0 - 0	4 - 0	21.4
SafeSCAD	0 - 0	8 - 0	2 - 0	**2** - 0	4 - **1**	42.4

We also study the resolution effort, please find the details in the paper

Conclusion

Goal: support non-technical stakeholders in eliciting well-formed normative requirements

Our contributions:

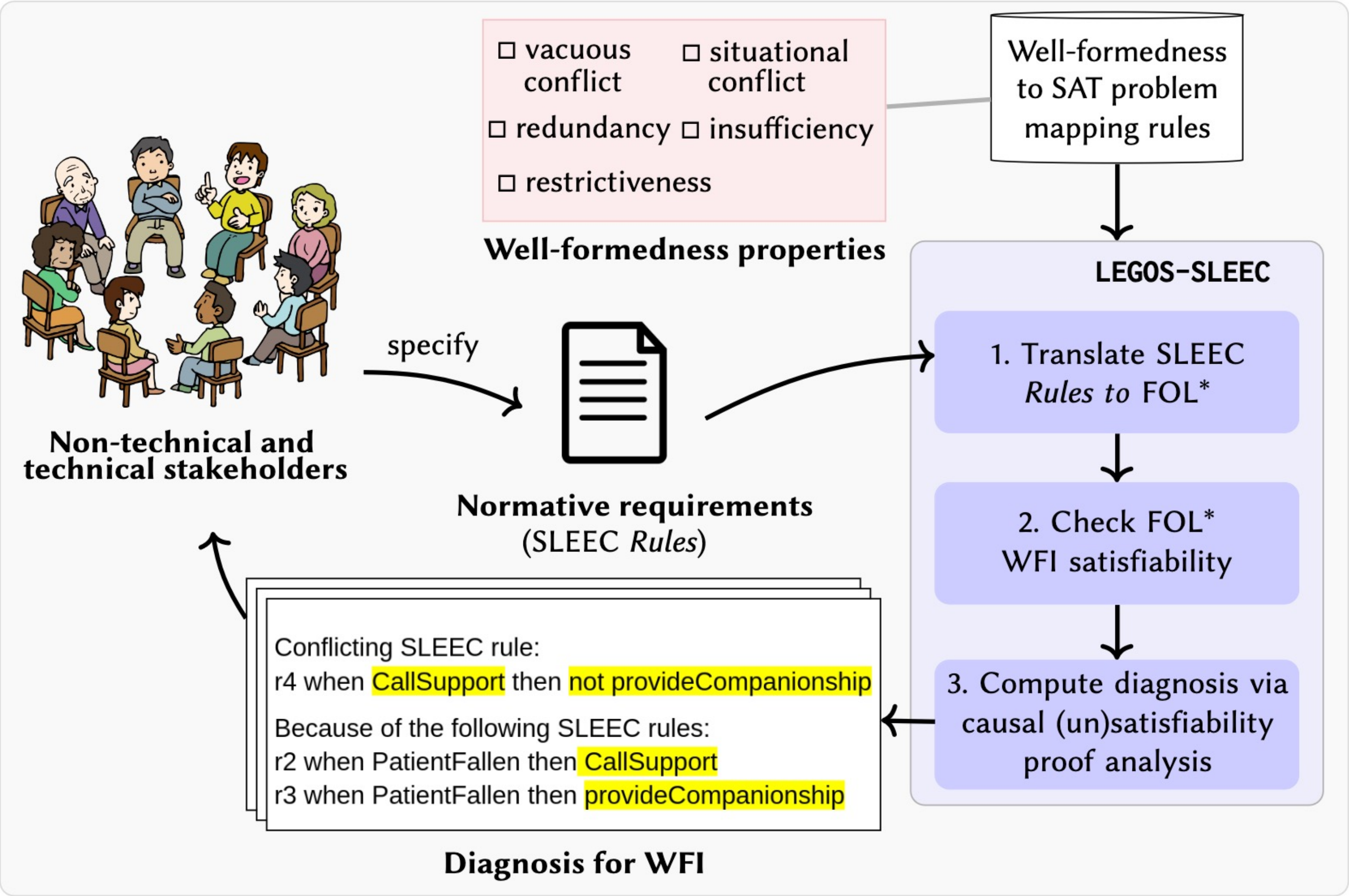
- Provided **automated verification** of five well-formdness properties
  1. situational conflict – 2. vacous conflict –
  3. insufficiency – 4. restrictiveness – 5. redundancy
- Developed **`readable' verification diagnosis**

Outcome: An effective engagement with a formal reasoning tool for non-technical stakeholders!

Future research directions

a) How to capture semantic relations between abstract representation of system capabilities with LLMs?
[Check out our upcoming RE 2024 paper]

b) How to assure that **systems** satisfy their SLEEC constraints:
- Via runtime monitoring
- Via formal verification
- Via synthesis of guardrails



An effective engagement with a formal reasoning tool for non-technical stakeholders!

Thank you!

Questions?

Tool:
LEGOS-SLEEC: https://github.com/NickF0211/LEGOS-SLEEC

Repository:

RESERVE: http://www.cs.toronto.edu/~sleec/