33rd Annual INCOSE international symposium

hybrid event

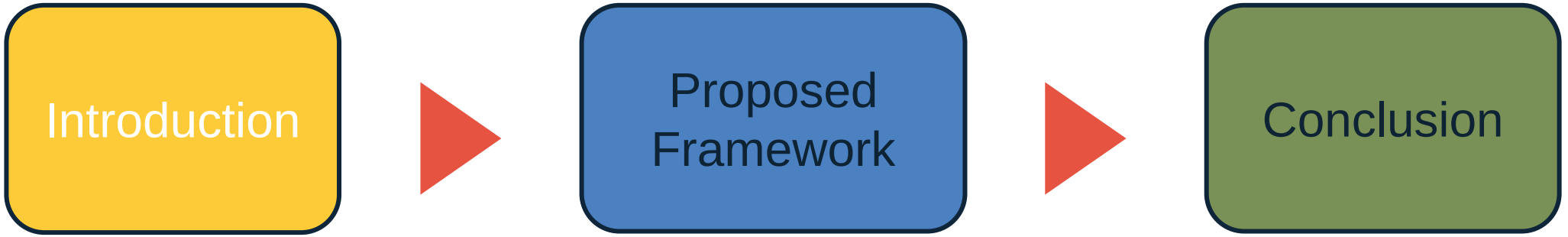
Honolulu, HI, USA
July 15 - 20, 2023

PURDUE UNIVERSITY

Ibukun Phillips & Robert Kenley – Purdue University

System Verification via Model-Checking: A Case study of a multi-differential drive robot

Presentation Outline

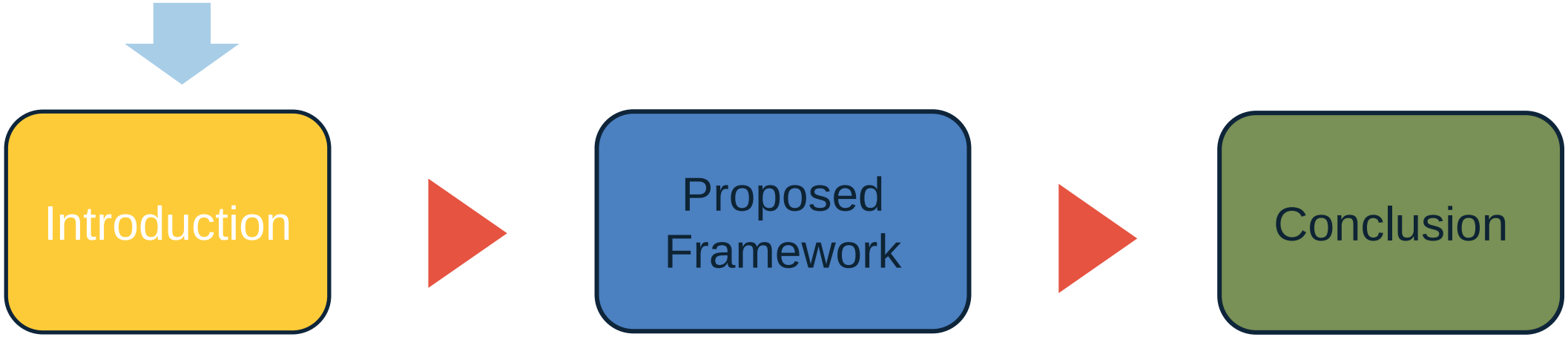


- MBSE/Cyber-physical systems
- Verification in SE
- Formal Methods
- Model Checking
- Temporal Logic Language

- NuSMV Model Checker
- Kripke Structure
- Autonomous multi-differential drive robot (DDR) Case Study
- Verification and Results

- Interpretation of Results
- Limitations & Future Work

Introduction

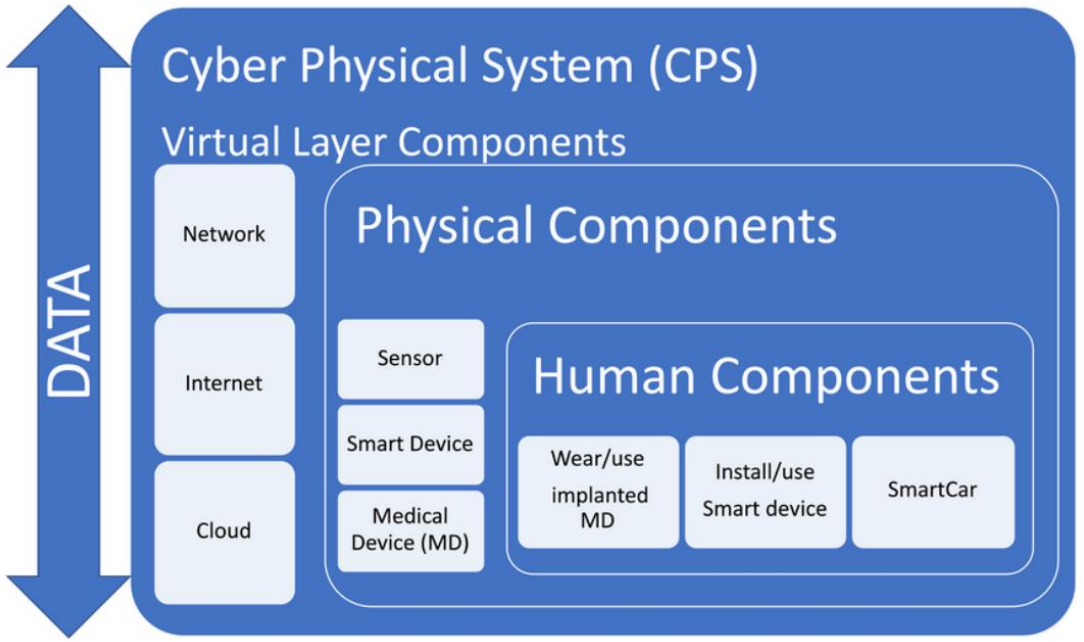


- MBSE/Cyber-physical systems
- Systems Verification in MBSE
- Formal Methods in SE
- Model Checking
- Temporal Logic Language

MBSE & Cyber-Physical Systems design

- What are Cyber-physical systems (CPS)?
  - Integration of physical, communication and computing technology systems
  - Complex multidisciplinary systems
  - Applications: Smart Manufacturing, Emergency Response, Air Transportation, Critical Infrastructure, Health Care and Medicine, Intelligent Transportation, etc.
  - CPS design entails a multidisciplinary approach (Lefèvre, et al., 2014)

Opportunities for Model-Based Systems Engineering (MBSE)



Cyber-physical system components.
Source: (Henriksen-Bulmer, et al (2000))

Systems Verification in MBSE

Pros

- MBSE supported with SysML & UML languages for system modeling
- Leverages its diagram types for the specification, analysis, design, verification, and validation of CPS.

Challenges

- Common modeling languages are semi-formal modeling approach and lacks mathematically rigorous, high-level formal language that can support automated design and analysis of systems (Wang, et al., 2019)
- Inapplicability of existing *conventional* verification approaches for rigorously evaluating these systems against system specifications
- Applications: systems where a design failure would result in a catastrophic situation such as hardware systems, traffic control software, aerospace systems etc.

Formal Methods in Systems Engineering

- Most common V&V (verification & validation) method in literature (Martínez-Fernández, 2022)
- Formal verification is broadly classified into deductive verification and ***model checking*** (Seger, 1992)
- Utilized to analyze and verify system (hardware and software) models at any part of the system lifecycle
- Formal methods + system models and SysML/UML diagrams → Effective formal modeling and specification of systems to ensure the correctness of system requirements.

Model Checking in SE Schools of Thought

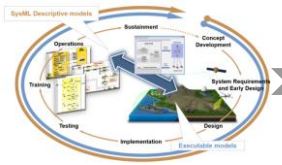
Mahani et al. (2021)

Direct

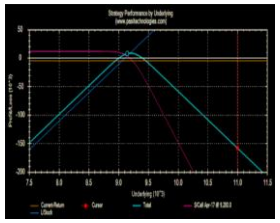
(Clarke & Heinle, 2000)

Indirect

(Caltais, et al., 2016; Kölbl, et al., 2018)



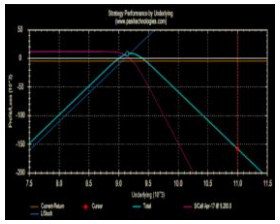
SysML model



Verification Tool

SysML model

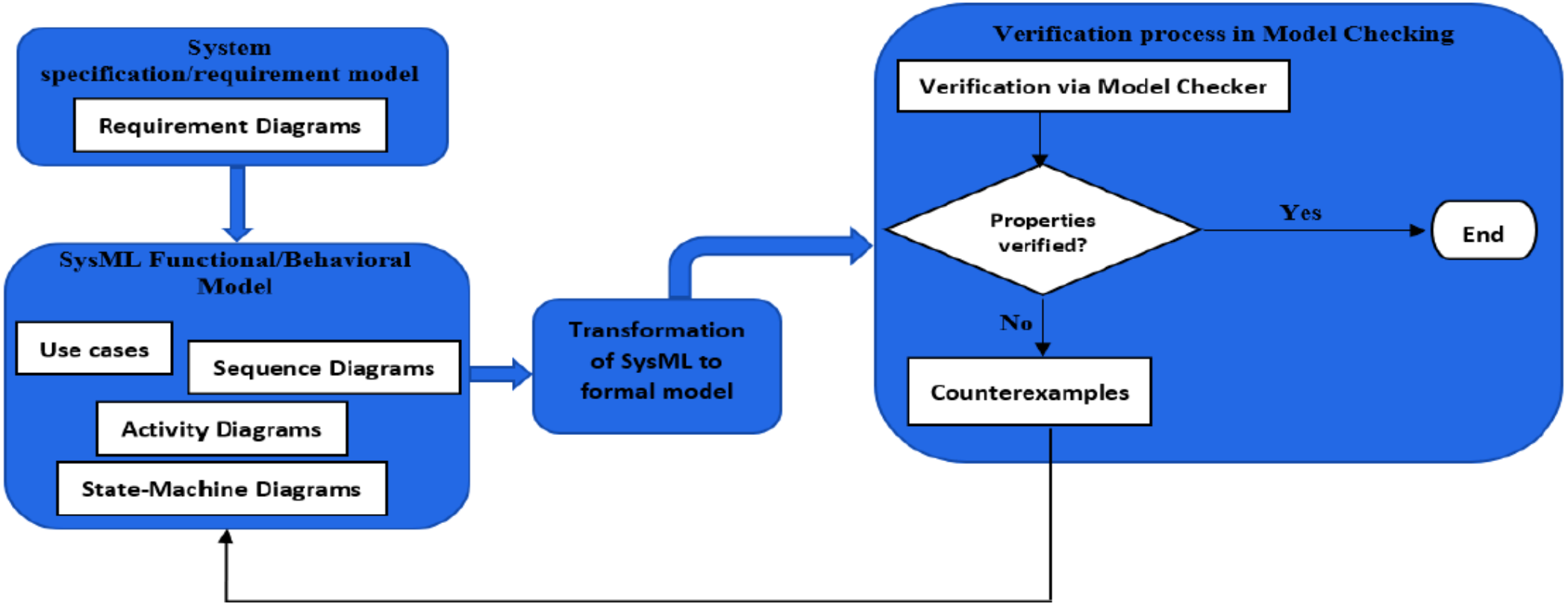
Intermediate model



Verification Tool

Formal Methods: Model Checking

- Indirect School of Thought for model checking application

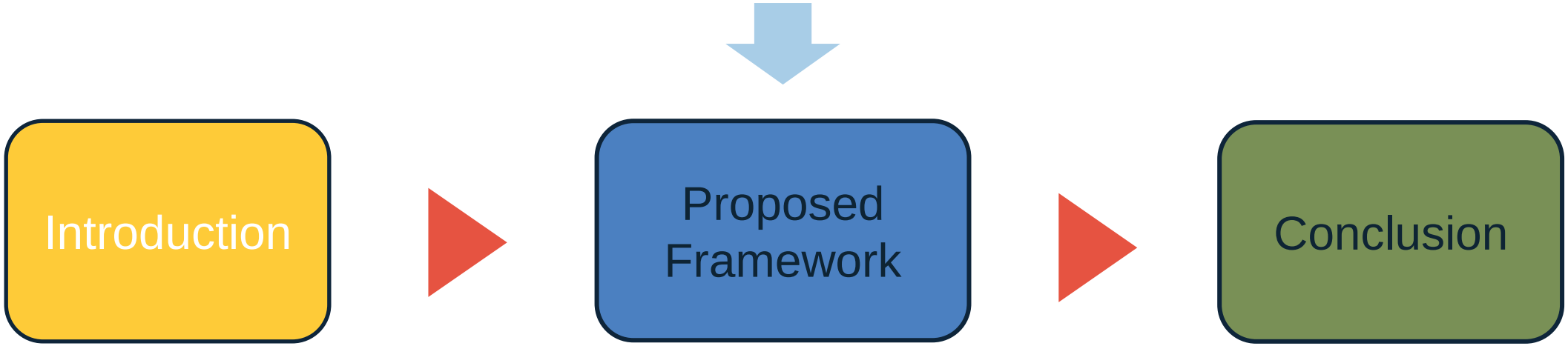


Model Checking Process leveraging on System Models

Model Checking: Temporal Logic Language

- TLL is the verification language for the formalism for this transformation to be used by model checkers.
- Classified into linear temporal logic (linear time structure) and Computational tree logic (branching time structure). (Lahtinen, 2008).
- Formal semantics of temporal formulas is defined for paths of a Kripke structure (Biere, et al., 2009) - crucial in expressing temporal issues in reactive and concurrent systems like CPS
- Typical Linear temporal logic operators are:
  - `F p` (read “in the future `p`”), stating that a certain condition `p` holds in one of the future time instants;
  - `G p` (read “globally `p`”), stating that a certain condition `p` holds in all future time instants;
  - `p U q` (read “`p` until `q`”), stating that condition `p` holds until a state is reached where condition `q` holds;
  - `X p` (read “next `p`”), stating that condition `p` is true in the next state.

Proposed Framework



- NuSMV Model Checker
- Kripke Structure
- Autonomous multi-differential drive robot (DDR) Case Study
- Verification and Results

NuSMV Model Checker

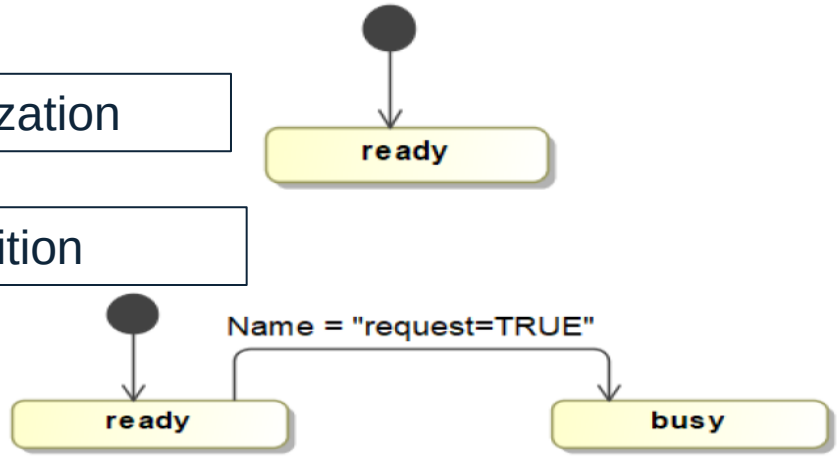
- New Symbolic Model Verifier (NuSMV)
  - Used to analyze temporal logic specifications of various systems (Cavada, et al., 2005)
  - Abstract notation of Kripke structure
  - Permits Linear Temporal Logic (LTL), Computational Tree Logic (CTL), and the Property Specification Language (PSL).

```
1 MODULE main
2
3    VAR
4    request : boolean; state : {ready,busy};
5
6    ASSIGN
7    init(state) := ready;
8    next(state) := case
9                   state = ready & request : busy;
10                                  TRUE : {ready,busy};
11                   esac;
```

variable declaration

variable initialization

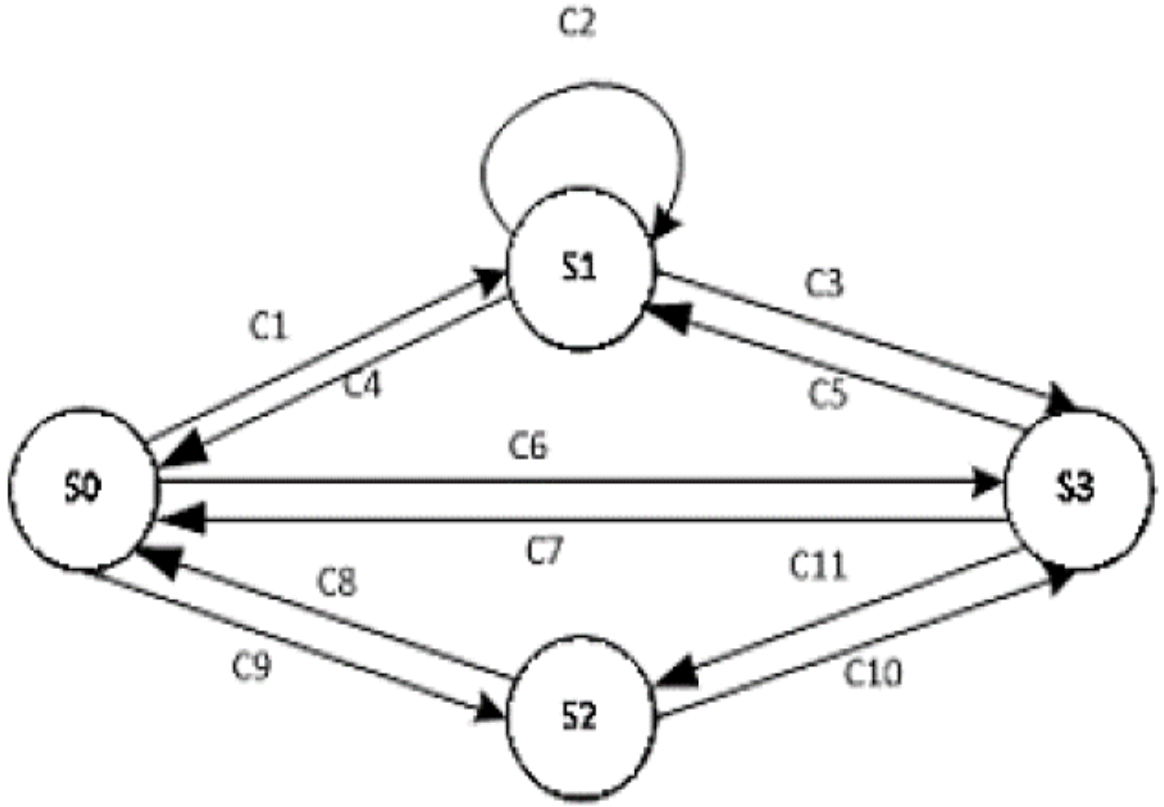
variable transition



A simple NuSMV Module declaration. Source: (Cavada, et al., 2013)

Kripke Structure in NuSMV from SysML Transformation

- Underlying abstraction of the model checking process based on the concept of Finite State Machines (FSMs)

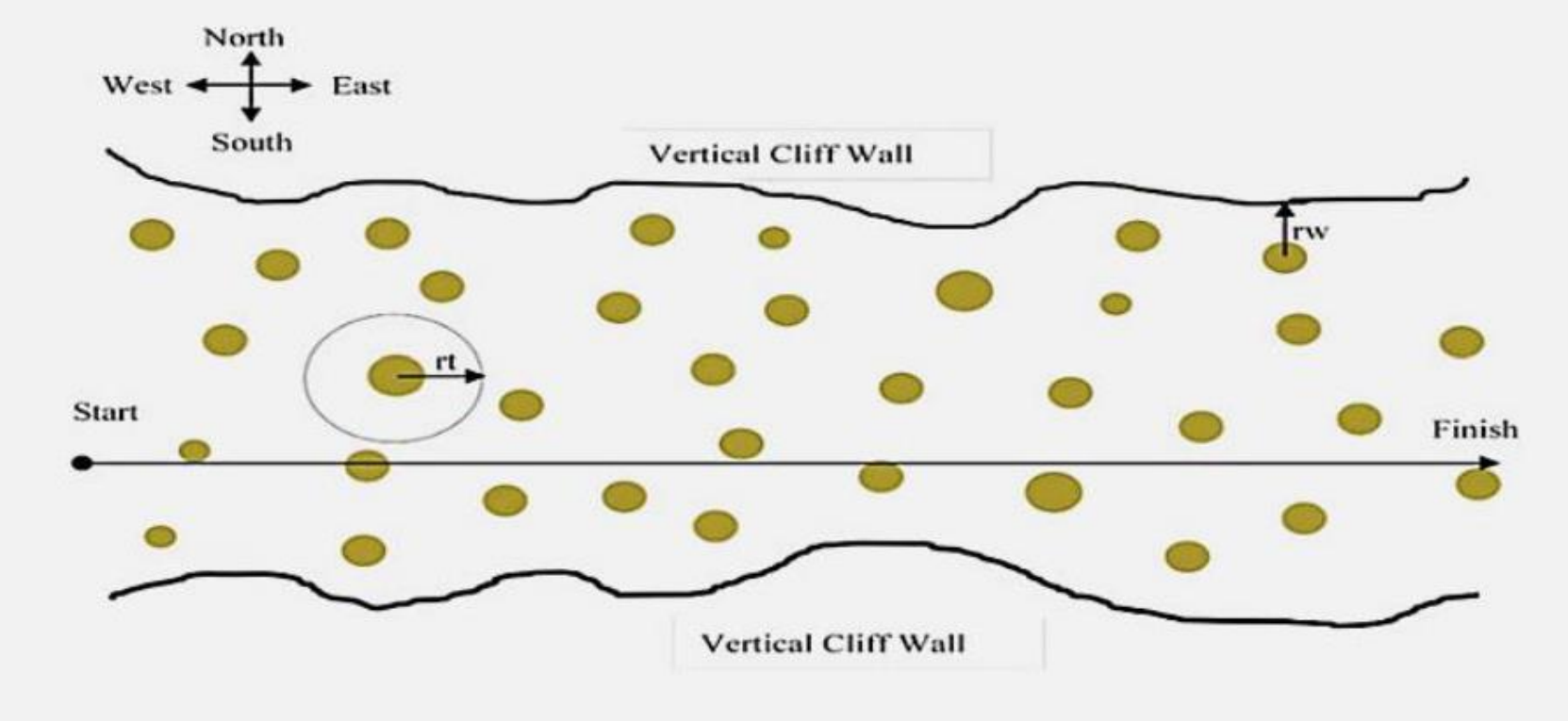


Finite State Machine model. Source: (Ding, et al., 2018)

- defined as a quadruple $M = (S, R, S_0, L)$ where S is the set of states with $S = \{S0, S1, S2, S3\}$
- $S_0 \subseteq S$ is the set of initial states
- $R \subseteq S \; X \; S$ is the transition relation with $C = \{C1, \ldots, C11\}$ being the transition set
- and $L: S \rightarrow P(A)$ is the labeling function, where A is the set of atomic propositions, and $P(A)$ is the powerset over A

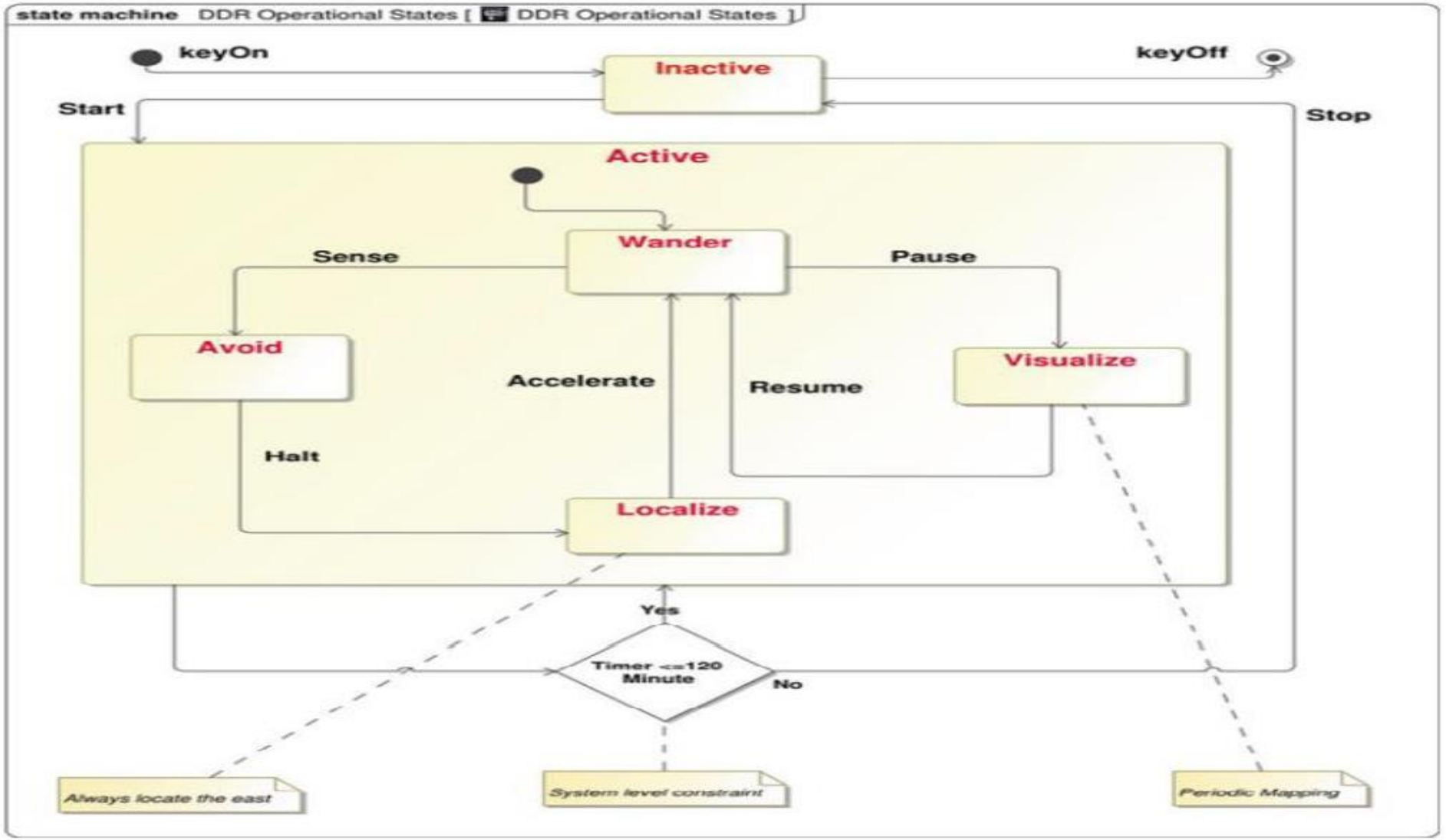
Case study: Autonomous multi-differential drive robot (DDR)

- DDR system is a CPS of autonomous mobile robots (differential drive robots)
  - developed to identify and retrieve a target inside a forest with an unknown path plan
- Goal: utilize model checking to verify some system models already available for this DDR system that can be described in terms of the forest environment, operational specifications, and differential robot specifications.



System Environment Diagram for the DDR. Source: (Abu Al-Haija, 2022)

Model Transformation: DDR Operational States



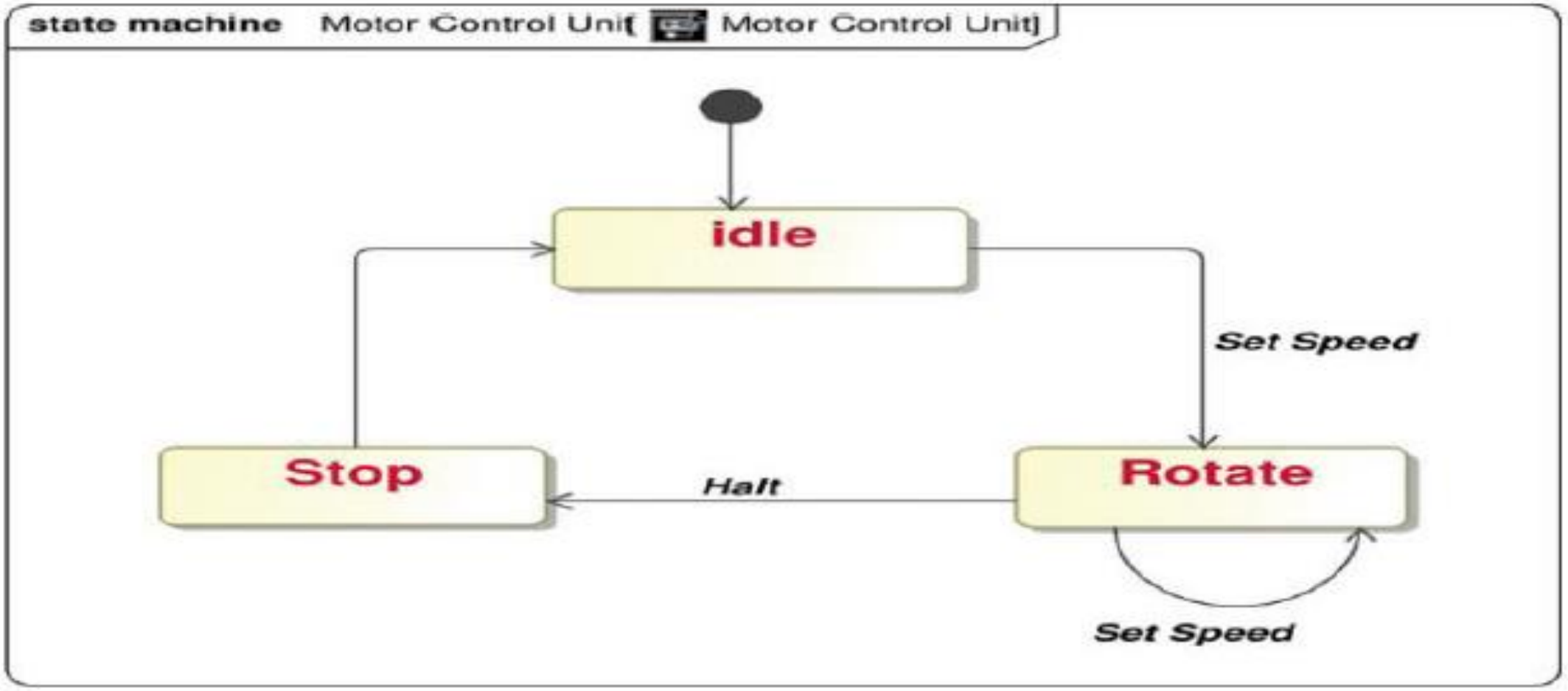
STM Diagram for the DDR System Operational States. Source: (Abu Al-Haija, 2022)

Model Transformation: DDR Operational States

```
MODULE active
        VAR
                sense           : boolean;
                pause           : boolean;
                halt            : boolean;
                accelerate      : boolean;
                resume          : boolean;
                active_state    : {wonder, avoid, localize, visualize};
        ASSIGN
                init (active_state) := wander;
                next (active_state) := case
                                active_state = wander & sense                   : avoid;
                                active_state = wander & pause                   : visualize;
                                active_state = avoid & halt                     : localize;
                                active_state = localize & accelerate            : wander;
                                active_state = visualize & resume               : wander;
                                TRUE                                            : active_state;
                                esac;
MODULE main
        VAR
           keyOff: boolean;
           start: boolean;
           state: {inactive, active, off};
           timer: 1..120;
        ASSIGN
                init (state) := inactive;
                init (timer) := 1;
                next (state) := case
                                state = inactive & start                                : active;
                                (state = active) & (timer <= 120)                       : active;
                                (state = active) & (timer > 120)                        : inactive;
                                state = inactive & keyOff                               : off;
                                TRUE                                                    : state;
                                esac;
```

NuSMV model for the DDR Operational States

Model Transformation: Motor Control Unit



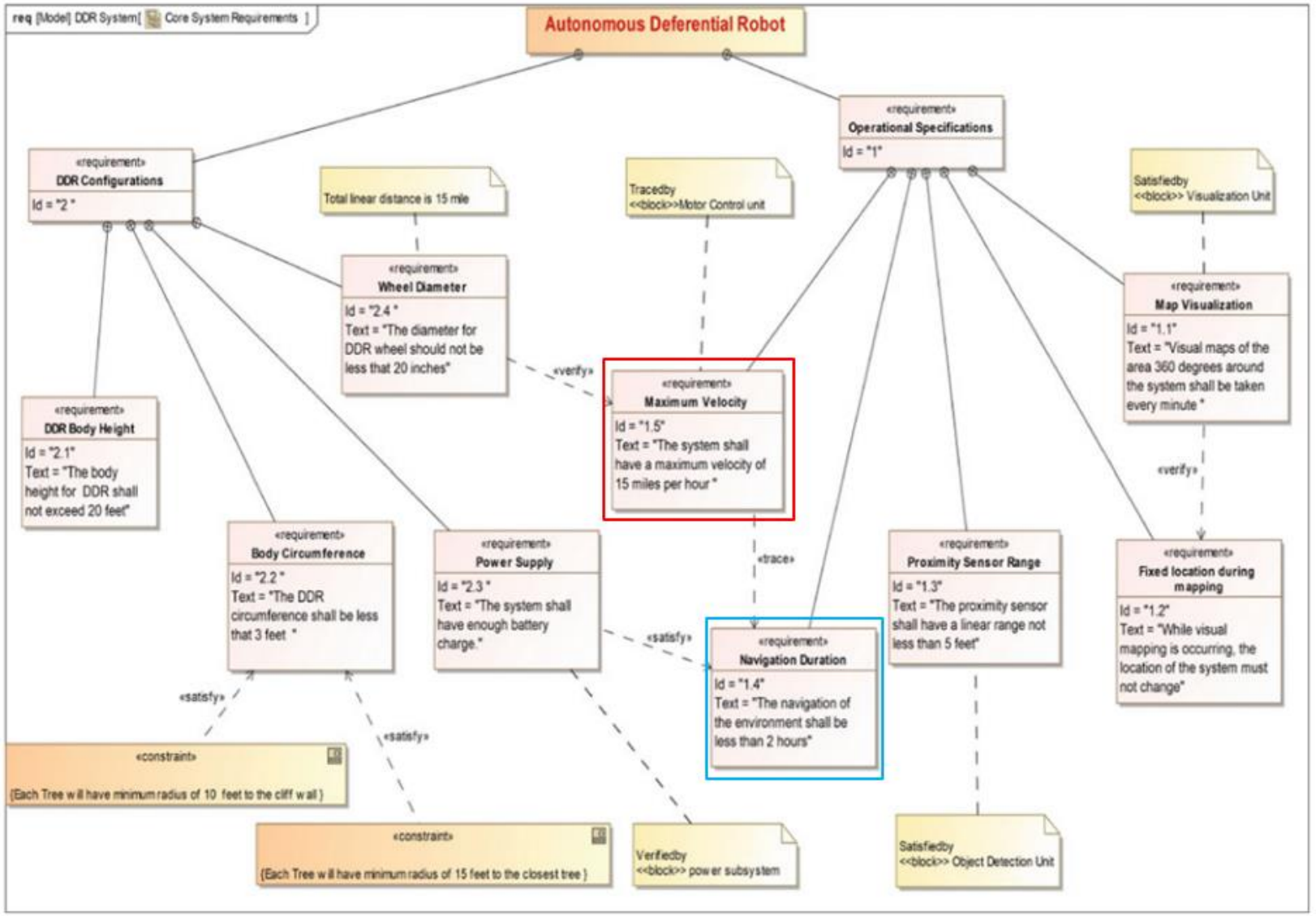
STM Diagram for the DDR Motor Control Unit. Source: (Abu Al-Haija, 2022)

Model Transformation: Motor Control Unit

```
MODULE main
        VAR
           set_speed: 1..15;
           halt: boolean;
           no_navigation: boolean;
           state: {idle, rotate, stop};

        ASSIGN
                init (state) := idle;
                next (state) := case
                                state = idle & set_speed<=15          : rotate;
                                state = rotate & set_speed<=15        : rotate;
                                state = rotate & halt                 : stop;
                                state = stop & no_navigation          : idle;
                                TRUE                                  : state;
                                esac;
```

NuSMV model of the Motor Control Unit

Verification and Results



System Requirement Diagram for the Autonomous DDR. Source: (Abu Al-Haija, 2022)

DDR Operational States model

Requirement:

- LTLSPEC -p “G (state = active & timer <= 120)”

Motor Control Unit model

Requirement:

- LTLSPEC -p “G (state = rotate & set_speed <= 15)”

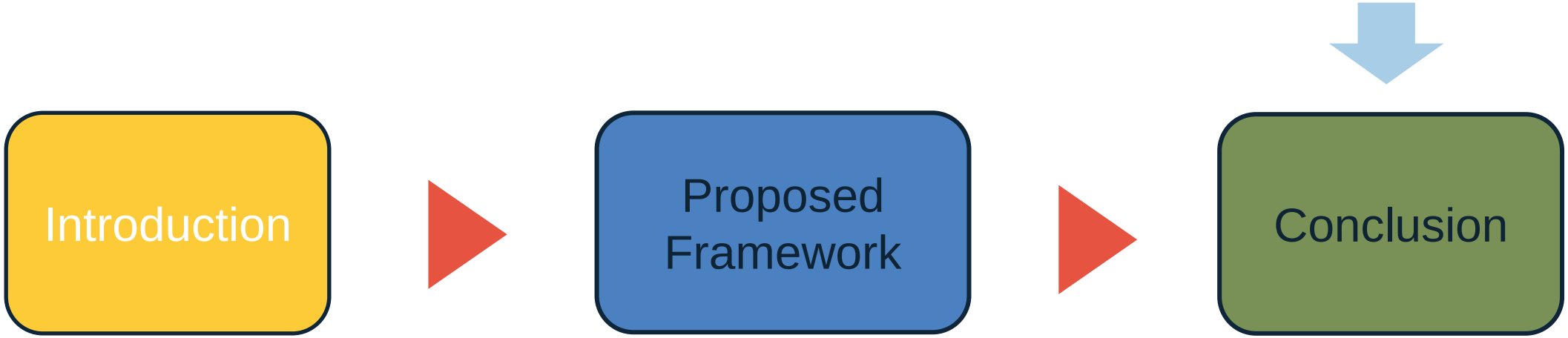
Verification and Results

```
*** Copyright (c) 2010-2014, Fondazione Bruno Kessler

*** This version of NuSMV is linked to the CUDD library version 2.4.1
*** Copyright (c) 1995-2004, Regents of the University of Colorado

*** This version of NuSMV is linked to the MiniSat SAT solver.
*** See http://minisat.se/MiniSat.html
*** Copyright (c) 2003-2006, Niklas Een, Niklas Sorensson
*** Copyright (c) 2007-2010, Niklas Sorensson

NuSMV > read_model -i ddr_Ops.smv
NuSMV > flatten_hierarchy
NuSMV > encode_variables
NuSMV > build_model
NuSMV > check_ltlspec -p "G(state=active & timer<=120)"
-- specification  G (state = active & timer <= 120)  is false
-- as demonstrated by the following execution sequence
Trace Description: LTL Counterexample
Trace Type: Counterexample
  -> State: 1.1 <-
    keyOff = FALSE
    start = TRUE
    state = inactive
    timer = 1
  -- Loop starts here
  -> State: 1.2 <-
    start = FALSE
    state = active
  -> State: 1.3 <-
NuSMV >
```

```
*** Copyright (c) 2010-2014, Fondazione Bruno Kessler

*** This version of NuSMV is linked to the CUDD library version 2.4.1
*** Copyright (c) 1995-2004, Regents of the University of Colorado

*** This version of NuSMV is linked to the MiniSat SAT solver.
*** See http://minisat.se/MiniSat.html
*** Copyright (c) 2003-2006, Niklas Een, Niklas Sorensson
*** Copyright (c) 2007-2010, Niklas Sorensson

NuSMV > read_model -i MCU.smv
NuSMV > flatten_hierarchy
NuSMV > encode_variables
NuSMV > build_model
NuSMV > check_ltlspec -p "G(set_speed<=15)"
-- specification  G set_speed <= 15  is true
NuSMV > check_ltlspec -p "G(state=rotate & set_speed<=15)"
-- specification  G (state = rotate & set_speed <= 15)  is false
-- as demonstrated by the following execution sequence
Trace Description: LTL Counterexample
Trace Type: Counterexample
  -> State: 1.1 <-
    set_speed = 1
    halt = FALSE
    no_navigation = FALSE
    state = idle
  -- Loop starts here
  -> State: 1.2 <-
    state = rotate
  -> State: 1.3 <-
```

Verification Results

Conclusion



- Interpretation of Results
- Limitations & Future Work

Interpretation of Results

- For both models, the specifications are false, which means that the NuSMV models do not satisfy the requirements defined in LTL.
- Conversely, a counterexample is provided in both models to detail the execution sequence that breaks the LTL specification/requirement formula.
- An implication of the false specification for the DDR navigation requirement in State 1.1 is that the DDR could be in an inactive phase. At the same time, the timer variable is at most 120 minutes which shows that the navigation duration requirement is unrealistic. In reality, the timer can be at most 120 minutes, but the DDR may not be fully operational and switched on for it to be in the active phase
- The model checking process helped identify the faulty requirements in the SysML requirement diagram of the DDR
- Defects in product/system development and verification, especially in system design, can be found using the proposed method.

Limitations & Future Work

- State-space explosion problem
  - Binary Decision Diagrams (BDDs)
  - Abstraction
  - Partial-order reduction (Lahtinen, 2008)

Future Work

- Other behavioral SysML diagrams (use case, activity, and sequence diagrams) can be formally expressed in the verification language of a model checker like NuSMV

- Partial or full automation of the SysML model transformation into the model checking tool (NuSMV).

References

- Abu Al-Haija, Q., 2022. *SysML-Based Design of Autonomous Multi-robot Cyber-Physical System Using Smart IoT Modules: A Case Study*. Machine Learning Techniques for Smart City Applications: Trends and Solutions, pp. 203-219.
- Biere, A. et al., 2009. Bounded model checking. In: Handbook of satisfiability., pp. 457-481.
- Caltais, G., Leitner-Fischer, F., Leue, S. & Weiser, J., 2016. SysML to NuSMV model transformation via object-orientation. Cham, Springer, pp. 31-45.
- Cavada, R. et al., 2005. *Nusmv 2.4 user manual*.
- Cavada, R. et al., 2013. *NuSMV 2.5 User Manual*.
- Clarke, E. M., & Heinle, W. (2000). *Modular translation of Statecharts to SMV. Technical Report CMU-CS-00-XXX*, Carnegie Mellon University School of Computer Science.
- Ding, Y. et al., 2018. *System states transition safety analysis method based on FSM and NuSMV*, pp. 107-112.
- Lahtinen, J., 2008. *Simplification of NuSMV Model Checking Counter Examples*.
- Lefèvre, J. et al., 2014. *Multidisciplinary modelling and simulation for mechatronic design.* Journal of Design Research, 9(12), pp. 127-144.
- Kölbl, M., Leue, S. & Singh, H., 2018. From SysML to model checkers via model transformation. Cham, Springer, pp. 255-274.
- Martínez-Fernández, Bogner, J., Franch, X., Oriol, M., Siebert, J., Trendowicz, A., Vollmer, A.M., and Stefan Wagner., 2022. *Software engineering for AI-based systems: a survey.. ACM Transactions on Software Engineering and Methodology*, 31(2), pp. 1-59.
- Mahani, M., Rizzo, D., Paredis, C. & Wang, Y., 2021. Automatic formal verification of SysML state machine diagrams for vehicular control system, : SAE.
- Henriksen-Bulmer, J., Faily, S., & Jeary, S. (2020). *DPIA in context: applying dpia to assess privacy risks of cyber physical systems*. Future internet, 12(5), 93.

33rd Annual INCOSE international symposium

hybrid event

Honolulu, HI, USA
July 15 - 20, 2023

www.incose.org/symp2023
#INCOSEIS