

# Formal verification of systems – a survey of approaches from classical to recent developments

Prof. Dr.-Ing. Sebastian Schlesinger

Berlin School for Economics and Law

June 14, 2024

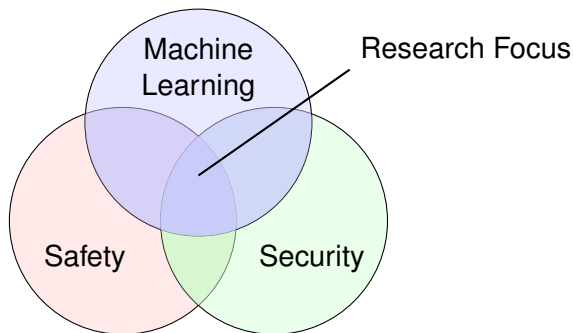
# Objectives

- Obtain an initial understanding of formal concepts
- Survey of classical and recent approaches to formal verification
- Also establish the bridge to related work and future research directions I am aiming at

# My Research Focus

My background: **formal verification** (particularly model-driven engineering of embedded or cyber-physical systems) and **security**. Recently, also **machine learning**.

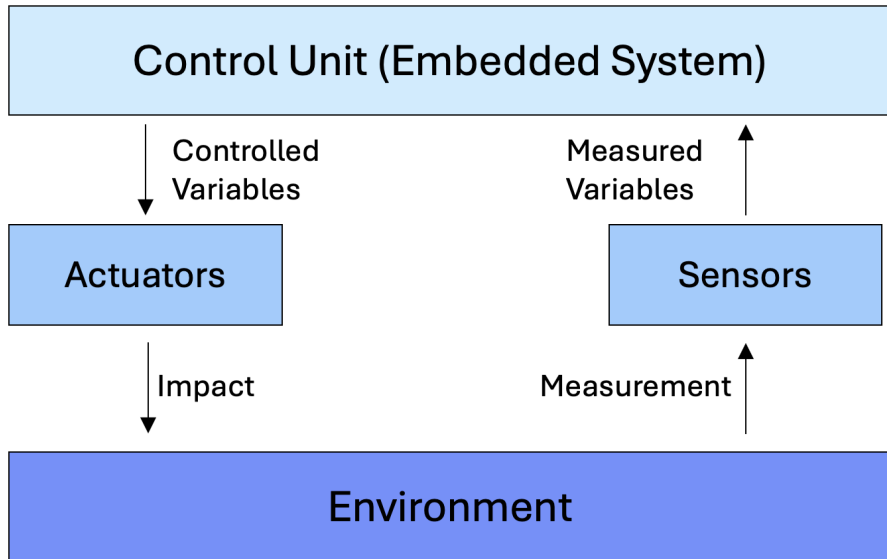
So, in essence, I am interested in **safety** and **security** of **AI-enabled systems** or the application of **Machine Learning** to classical approaches for the verification of safety and security of systems.



# Outline

- 1 Introduction
- 2 First-order logic
- 3 Verification of sequential systems

# Embedded Systems



# Why formal verification?

# Language of first-order logic

A language  $\mathcal{L}$  of first-order logic consists of the following components:

- Variable symbols:  $x_1, x_2, \dots$
- For each  $n \in \mathbb{N}$ , a set of  $n$ -ary function symbols:  $f_0, f_1, \dots$ . The 0-ary function symbols are called constant symbols.
- For each  $n \in \mathbb{N}$ , a set of  $n$ -ary predicate symbols:  $p_0, p_1, \dots$ . The 0-ary predicate symbols are the constants  $\top$  (for **true**) and  $\perp$  (for **false**).
- special symbols:  $\neg$  (negation),  $\wedge$  (conjunction),  $\vee$  (disjunction),  $\rightarrow$  (implication),  $\leftrightarrow$  (equivalence),  $\forall$  (universal quantification),  $\exists$  (existential quantification), and parentheses.

# Terms

The set of terms of  $\mathcal{L}$  is defined inductively as follows:

- Each variable is a term.
- If  $t_1, \dots, t_n$  are terms and  $f$  is an  $n$ -ary function symbol, then if  $f(t_1, \dots, t_n)$  is a term.



# Variables in terms

We define a function  $var : \text{Terms} \rightarrow \text{Variables}$  that maps each term to the set of variables occurring in it. The function is defined as follows:

- $var(x) = \{x\}$  for each variable  $x$ .
- $var(f(t_1, \dots, t_n)) = var(t_1) \cup \dots \cup var(t_n)$ .

# Formulas

The set of formulas of  $\mathcal{L}$  is defined inductively as follows:

- If  $t_1, \dots, t_n$  are terms and  $p$  is an  $n$ -ary predicate symbol, then if  $p(t_1, \dots, t_n)$  is a formula.
- If  $\varphi$  is a formula, then if  $\neg\varphi$  is a formula.
- If  $\varphi_1$  and  $\varphi_2$  are formulas, then if  $\varphi_1 \wedge \varphi_2$ ,  $\varphi_1 \vee \varphi_2$ ,  $\varphi_1 \rightarrow \varphi_2$ , and  $\varphi_1 \leftrightarrow \varphi_2$  are formulas.
- If  $\varphi$  is a formula and  $x$  is a variable, then if  $\forall x.\varphi$  and  $\exists x.\varphi$  are formulas.

An example of a formula is  $\forall x.\exists y.p(x, y) \rightarrow \neg q(y)$ .

# Interpretations

An interpretation  $\mathcal{M}$  of  $\mathcal{L}$  consists of the following components:

- A non-empty set  $D$  called the domain of  $\mathcal{M}$ .
- For each  $n$ -ary function symbol  $f$  of  $\mathcal{L}$ , a function  $f^{\mathcal{M}} : D^n \rightarrow D$ .
- For each  $n$ -ary predicate symbol  $p$  of  $\mathcal{L}$ , a relation  $p^{\mathcal{M}} \subseteq D^n$ .

# Interpretations of Terms

Let  $\mathcal{M}$  be an interpretation for our first-order language. An assignment  $\sigma$  of values to variables, i.e.,  $\sigma : Variables \rightarrow D$ .

The value of a term  $t$  under  $\sigma$  is denoted by  $t^{\mathcal{M}}[\sigma]$  and defined as follows:

- If  $t = x$  for a variable  $x$ , then  $t^{\mathcal{M}}[\sigma] = \sigma(x)$ .
- If  $t = f(t_1, \dots, t_n)$ , then  $t^{\mathcal{M}}[\sigma] = f^{\mathcal{M}}(t_1^{\mathcal{M}}[\sigma], \dots, t_n^{\mathcal{M}}[\sigma])$ .

# Validity of Formulas under Interpretations

We say an assignment  $\sigma$  satisfies a formula  $\varphi$  under an interpretation  $\mathcal{M}$ , denoted by  $\mathcal{M}, \sigma \models \varphi$ , iff the following conditions hold:

- $\varphi = p(t_1, \dots, t_n)$ , then if  $(t_1^{\mathcal{M}}[\sigma], \dots, t_n^{\mathcal{M}}[\sigma]) \in p^{\mathcal{M}}$ .
- $\varphi = \neg\psi$ , then if  $\mathcal{M}, \sigma \not\models \psi$ .
- $\varphi = \psi_1 \vee \psi_2$ , then if  $\mathcal{M}, \sigma \models \psi_1$  or  $\mathcal{M}, \sigma \models \psi_2$ .
- $\varphi = \psi_1 \wedge \psi_2$ , then if  $\mathcal{M}, \sigma \models \psi_1$  and  $\mathcal{M}, \sigma \models \psi_2$ .
- $\varphi = \psi_1 \rightarrow \psi_2$ , then if  $\mathcal{M}, \sigma \models \psi_1$  implies  $\mathcal{M}, \sigma \models \psi_2$ .
- $\varphi = \psi_1 \leftrightarrow \psi_2$ , then if  $\mathcal{M}, \sigma \models \psi_1$  if and only if  $\mathcal{M}, \sigma \models \psi_2$ .
- $\varphi = \forall x.\psi$ , then if  $\mathcal{M}, \sigma[x \mapsto d] \models \psi$  for all  $d \in D$ .
- $\varphi = \exists x.\psi$ , then if  $\mathcal{M}, \sigma[x \mapsto d] \models \psi$  for some  $d \in D$ .

A formula  $\varphi$  is satisfiable if there exists an interpretation  $\mathcal{M}$  and an assignment  $\sigma$  such that  $\mathcal{M}, \sigma \models \varphi$ .

# Models

An interpretation  $\mathcal{M}$  is a model of a formula  $\varphi$ , denoted by  $\mathcal{M} \models \varphi$ , if for all assignments  $\sigma$ ,  $\mathcal{M}, \sigma \models \varphi$ .

A formula is satisfiable if it has a model, i.e., if there exists an interpretation  $\mathcal{M}$  such that  $\mathcal{M} \models \varphi$ .

# Validity

A formula  $\varphi$  is valid if for all interpretations  $\mathcal{M}$  and all assignments  $\sigma$ ,  $\mathcal{M}, \sigma \models \varphi$ .

We write  $\models \varphi$  to denote that  $\varphi$  is valid.

# Free Variables in Formulas

The set of free variables of a formula  $\varphi$ , denoted by  $FV(\varphi)$ , is defined inductively as follows:

- $FV(p(t_1, \dots, t_n)) = \text{var}(t_1) \cup \dots \cup \text{var}(t_n)$ .
- $FV(\neg\psi) = FV(\psi)$ .
- $FV(\psi_1 \wedge \psi_2) = FV(\psi_1) \cup FV(\psi_2)$ .
- $FV(\psi_1 \vee \psi_2) = FV(\psi_1) \cup FV(\psi_2)$ .
- $FV(\psi_1 \rightarrow \psi_2) = FV(\psi_1) \cup FV(\psi_2)$ .
- $FV(\forall x.\psi) = FV(\psi) \setminus \{x\}$ .
- $FV(\exists x.\psi) = FV(\psi) \setminus \{x\}$ .



# Term Substitution

Let  $\varphi$  be a formula,  $x$  a variable, and  $t$  a term. The formula  $\varphi[t/x]$  is obtained by replacing all occurrences of  $x$  in  $\varphi$  by  $t$ . The substitution is defined inductively as follows:

- $(p(t_1, \dots, t_n))[t/x] = p(t_1[t/x], \dots, t_n[t/x])$ .
- $(\neg\psi)[t/x] = \neg\psi[t/x]$ .
- $(\psi_1 \wedge \psi_2)[t/x] = \psi_1[t/x] \wedge \psi_2[t/x]$ .
- $(\psi_1 \vee \psi_2)[t/x] = \psi_1[t/x] \vee \psi_2[t/x]$ .
- $(\psi_1 \rightarrow \psi_2)[t/x] = \psi_1[t/x] \rightarrow \psi_2[t/x]$ .
- $(\forall y.\psi)[t/x] = \forall y.\psi[t/x]$  if  $x \in FV(t)$ .
- $(\exists y.\psi)[t/x] = \exists y.\psi[t/x]$  if  $x \in FV(t)$ .
- $(\forall x.\psi)[t/x] = \forall x.\psi$ .
- $(\exists x.\psi)[t/x] = \exists x.\psi$ .

So,  $\varphi[t/x]$  represents the formula obtained by substituting every **free** occurrence of the variable  $x$  in  $\varphi$  by the term  $t$ .

# Calculus

A calculus is a mechanism to prove formulas by applying rules.

A rule of a calculus has the form  $\frac{\varphi_1, \dots, \varphi_n}{\psi}$ , where  $\varphi_1, \dots, \varphi_n$  are premises and  $\psi$  is the conclusion. The rule states that if  $\varphi_1, \dots, \varphi_n$  are derivable, then  $\psi$  is derivable.

We denote that a formula can be proved by a calculus by  $\vdash \varphi$ .

# Sequent Calculus

In sequent calculus, we have sequences  $\Gamma \vdash \Delta$ , where  $\Gamma$  and  $\Delta$  are sets of formulas.

The interpretation is that if all formulas in  $\Gamma$  are true, then at least one formula in  $\Delta$  is true.

# Sequent Calculus Rules

$$\frac{-}{\Gamma, \varphi \Rightarrow \varphi, \Delta} \text{ Taut}$$

$$\frac{\Gamma \Rightarrow \Delta, \varphi \quad \varphi, \Pi \Rightarrow \Lambda}{\Gamma, \Pi \Rightarrow \Delta, \Lambda} \text{ Cut}$$

$$\frac{-}{\Gamma, \perp \Rightarrow \Delta} \perp \Rightarrow$$

$$\frac{-}{\Gamma \Rightarrow \Delta, \top} \Rightarrow \top$$

$$\frac{\Gamma \Rightarrow \Delta}{\varphi, \Gamma \Rightarrow \Delta} \text{ Weakening left}$$

$$\frac{\Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \varphi} \text{ Weakening right}$$

$$\frac{\varphi, \varphi, \Gamma \Rightarrow \Delta}{\varphi, \Gamma \Rightarrow \Delta} \text{ Contraction left}$$

$$\frac{\Gamma \Rightarrow \Delta, \varphi, \varphi}{\Gamma \Rightarrow \Delta, \varphi} \text{ Contraction right}$$

$$\frac{\Gamma, \varphi, \psi, \Pi \Rightarrow \Delta}{\Gamma, \psi, \varphi, \Pi \Rightarrow \Delta} \text{ Exchange left}$$

$$\frac{\Gamma \Rightarrow \Delta, \varphi, \psi, \Lambda}{\Gamma \Rightarrow \Delta, \psi, \varphi, \Lambda} \text{ Exchange right}$$

# Sequent Calculus Rules

$$\frac{}{\Gamma, \perp \Rightarrow \Delta} \perp \Rightarrow$$

$$\frac{\Gamma \Rightarrow \Delta, \varphi}{\Gamma, \neg \varphi \Rightarrow \Delta} \neg \Rightarrow$$

$$\frac{\Gamma, \varphi \Rightarrow \Delta \quad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \varphi \vee \psi \Rightarrow \Delta} \vee \Rightarrow$$

$$\frac{\Gamma, \varphi, \psi \Rightarrow \Delta}{\Gamma, \varphi \wedge \psi \Rightarrow \Delta} \wedge \Rightarrow$$

$$\frac{\Gamma \Rightarrow \Delta, \varphi \quad \psi, \Pi \Rightarrow \Lambda}{\varphi \rightarrow \psi, \Gamma, \Pi \Rightarrow \Delta, \Lambda} \rightarrow \Rightarrow$$

$$\frac{}{\Gamma \Rightarrow \Delta, \top} \top$$

$$\frac{\Gamma, \varphi \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \neg \varphi} \neg \Rightarrow$$

$$\frac{\Gamma \Rightarrow \Delta, \varphi, \psi}{\Gamma \Rightarrow \Delta, \varphi \vee \psi} \vee \Rightarrow$$

$$\frac{\Gamma \Rightarrow \Delta, \varphi \quad \Gamma \Rightarrow \Delta, \psi}{\Gamma \Rightarrow \Delta, \varphi \wedge \psi} \wedge \Rightarrow$$

$$\frac{\Gamma, \varphi \Rightarrow \Delta, \psi}{\Gamma \Rightarrow \Delta, \varphi \rightarrow \psi} \rightarrow \Rightarrow$$

# Sequent Calculus Rules

$$\frac{\Gamma, \varphi[t/x] \Rightarrow \Delta}{\Gamma, \forall x. \varphi(x) \Rightarrow \Delta} \forall \Rightarrow$$

$$\frac{\Gamma \Rightarrow \Delta, \varphi[y/x]}{\Gamma \Rightarrow \Delta, \forall x. \varphi(x)} \Rightarrow \forall$$

$$\frac{\Gamma, \varphi[y/x] \Rightarrow \Delta}{\Gamma, \exists x. \varphi(x) \Rightarrow \Delta} \exists \Rightarrow$$

$$\frac{\Gamma \Rightarrow \Delta, \exists x. \varphi(x), \varphi[t/x]}{\Gamma \Rightarrow \Delta, \exists x. \varphi(x)} \Rightarrow \exists$$

In the quantifier rules,  $t$  is a term, and  $y$  is a 'fresh' variable, i.e., a variable that does not occur in  $\Gamma$ ,  $\Delta$ , or  $\varphi$ .

Alternatively, the rules can also be stated in the form

$$\frac{\Gamma \Rightarrow \Delta, \varphi}{\Gamma \Rightarrow \Delta, \forall x. \varphi(x)} \Rightarrow \forall$$

Here, it must be guaranteed that  $x$  is not free in any formula in  $\Gamma$  or  $\Delta$ . The existential formula can be handled similarly.

# Example Deduction: $\forall x.(P(x) \wedge Q \Rightarrow \forall x.P(x))$

$$\begin{array}{c}
 \frac{}{P(x), Q \Rightarrow P(x)} \text{Taut} \\
 \frac{}{P(x) \wedge Q \Rightarrow P(x)} \wedge \Rightarrow \\
 \frac{}{\forall x.(P(x) \wedge Q) \Rightarrow P(x))} \forall \Rightarrow \\
 \frac{}{\forall x.(P(x) \wedge Q) \Rightarrow \forall x.P(x)} \Rightarrow \forall
 \end{array}$$

Here,  $\forall \Rightarrow$  uses  $[x/x]$  as replacement, i.e., just the same free variable is taken.

# Example Deduction: $\forall x.(A \rightarrow B) \Rightarrow A \rightarrow \forall x.B$

$$\begin{array}{c}
 \frac{}{A \Rightarrow A, B} \text{Taut} \quad \frac{}{A, B \Rightarrow B} \text{Taut} \\
 \hline
 \frac{}{A, A \rightarrow B \Rightarrow B} \rightarrow \Rightarrow \\
 \hline
 \frac{}{A, \forall x.(A \rightarrow B) \Rightarrow B} \forall \Rightarrow \\
 \hline
 \frac{}{A, \forall x.(A \rightarrow B) \Rightarrow \forall x.B} \Rightarrow \forall \\
 \hline
 \frac{}{\forall x.(A \rightarrow B) \Rightarrow A \rightarrow \forall x.B} \Rightarrow \rightarrow
 \end{array}$$

Here, the application of  $\Rightarrow \forall$  requires that  $x$  is not free in  $A$ .



## Example of a Failing Deduction:

$$\exists x.P(x) \wedge \exists x.Q(x) \Rightarrow \exists x.(P(x) \wedge Q(x))$$

$$\frac{\frac{\frac{P(x), Q(y) \Rightarrow P(x) \wedge Q(x)}{P(x), Q(y) \Rightarrow \exists x.(P(x) \wedge Q(x))} \Rightarrow \exists}{P(x), \exists x.Q(x) \Rightarrow \exists x.(P(x) \wedge Q(x))} \exists \Rightarrow}{\exists x.P(x), \exists x.Q(x) \Rightarrow \exists x.(P(x) \wedge Q(x))} \exists \Rightarrow \wedge \Rightarrow$$

Here, the deduction fails because the variable  $x$  is not fresh in the application of  $\exists \Rightarrow$  and therefore the new variable  $y$  is introduced. However, then the deduction cannot be completed.

# Soundness and Completeness of Sequent Calculus

- A calculus is sound if all provable formulas are valid, denoted by  $\vdash \varphi \Rightarrow \models \varphi$ .
- A calculus is complete if all valid formulas are provable, denoted by  $\models \varphi \Rightarrow \vdash \varphi$ .

The sequent calculus is sound and complete for first-order logic, i.e.,

- $\vdash \Gamma \Rightarrow \Delta$ , then  $\models \Gamma \Rightarrow \Delta$ .
- $\models \Gamma \Rightarrow \Delta$ , then  $\vdash \Gamma \Rightarrow \Delta$ .

Proof of soundness by induction on the structure of the formulas. Proof of completeness by constructing a Herbrand model.

# Goedel's Incompleteness Theorem

- Goedel's first incompleteness theorem states that in any consistent formal system that is powerful enough to express arithmetic, there are true statements that cannot be proven.
- Goedel's second incompleteness theorem states that in any consistent formal system that is powerful enough to express arithmetic, the system cannot prove its own consistency.

# Rice's Theorem

- Rice's theorem states that for any non-trivial property of partial functions, i.e., a property that is not true for all partial functions or not true for none, there is no algorithm that can decide whether a given program has that property.
- A property is non-trivial if there are two partial functions that are computable and one has the property and the other does not.

# Halting Problem

The halting problem is the problem of determining, given a program and an input, whether the program will eventually halt when run with that input.

The halting problem is undecidable, i.e., there is no algorithm that can decide whether a given program halts on a given input.

# Some Conclusions

- There are properties of programs that cannot be decided by an algorithm.
- There are properties of programs that cannot be verified by a formal system.
- It is impossible to generally prove behavioural equivalence of programs.



# Semantics of programs

There are three main types of semantics for programs:

- Operational semantics: Describes the execution of programs.
- Denotational semantics: Describes the meaning of programs (as a mathematical mapping of states).
- Axiomatic semantics: Describes properties of programs.



# The while language

The while language is a simple imperative programming language with the following constructs:

- Assignment:  $x := E$ , where  $x$  is a variable and  $E$  is an expression.
- Sequence:  $S_1; S_2$ , where  $S_1$  and  $S_2$  are statements.
- Conditional: if  $B$  then  $S_1$  else  $S_2$  end, where  $B$  is a boolean expression and  $S_1$  and  $S_2$  are statements.
- Loop: while  $B$  do  $S$  end, where  $B$  is a boolean expression and  $S$  is a statement.

# Operational Semantics of while

The operational semantics of the while language is defined by a transition relation  $\rightarrow$  between states.

A state is a mapping from variables to values.

The transition relation is defined by inference rules of the form  $\langle S, \sigma \rangle \rightarrow \sigma'$ , where  $S$  is a statement,  $\sigma$  is a state, and  $\sigma'$  is the resulting state.

# Operational Semantics of while