Lecture 1: Deductive Verification of Reactive Systems

- Introduction to propositional, predicate and higher order logics

- Deductive Invariance Proofs

Cristina Seceleanu
MRTC, MdH

E-mail: cristina.seceleanu@mdh.se
Example: Generalized Railroad Crossing

- **System description**
  - $N$ parallel railroad tracks protected by a barrier and a barrier controller.
  - Three track regions: $I$ (intersection), $P$ (an interval preceding the intersection), and notHere (everywhere else).
  - Barrier: down, up, goingDown, and goingUp.
  - Initially all trains are outside $P$ (notHere).

- **Requirements**
  - *Safe* system (safety): the barrier is down if a train is passing $I$
  - *Useful* system (utility): the barrier is up outside the $P$ region
Track $i$ time

$\tau_i - \varepsilon_1$
\[ \tau_i - \varepsilon_1 \quad \tau_i \quad \nu_i \quad \nu_i + \varepsilon_2 \]
What is deductive verification?

- **Deductive formal verification**
  - uses axioms and/or **proof rules** to prove the correctness of a given model (involves **interactive theorem-proving**: not fully automated, hindered application)

- Uses **logic** & **deductive** reasoning to mathematically prove that a design implements a well-defined specification:
  - Theorem-proving of functional containment:
    \[ \forall t: R_{\geq 0} \cdot \forall \text{inputs } x(t): (R_{\geq 0} \rightarrow \text{Type}). \ Spec(x(t)) \subseteq \text{Design}(x(t)) \]

- Model-checking automatically verifies that a Design is a model of a Spec written as a logical formula

- For real-time systems, x is a timed variable, i.e. a real-valued variable, and \( \text{Spec}(x(t)) \) can contain timing constraints.
Essentials in formal verification

- The basic steps in formal verification:
  - Formally model the system
  - Formalize the specification
  - Prove that the model satisfies the spec

- But what formalism should be used?

- Some typical formalisms
  - Temporal logic (CTL, LTL, TCTL etc.)
  - Propositional logic, a.k.a. Boolean logic
  - First-order logic
  - Higher-order logic ...
GRC example: Modeling the Trains

- Trains component: Timed Automaton with pre-post conditions
- r: variables ranging over trains
Possible Train Property

- Parameters:
  - $\varepsilon_1$ – lower bound on the time Train enters $P$ until reaches $I$
  - $\varepsilon_2$ – upper bound on the time Train exits $P$ until exits $I$

- $\forall r . \ r.\text{status} = P \Rightarrow (\text{first(Enter } I(r)) + \varepsilon_2 - \varepsilon_1 = \text{last(Enter } I(r)))$

- Prove this deductively 😊
Why deductive verification?

- Verifying reactive systems is difficult: need to get insights

- Fully automatic techniques (model checking):
  - only for finite-state models (state-explosion!): limited scope

- Deductive Verification enables proofs on infinite-state systems

- The deductive approach requires:
  - a general framework to reason about mathematics in general while checking against errors.
  - the use of an expressive (e.g., be able to express un-boundedness: \( \mathbb{N}, \mathbb{Z} \ldots \)) modeling framework and targeted proof techniques for
    - checking a proof
    - generating a proof
No free lunch

- There is no practical way of automatically proving highly sophisticated mathematics.

- Some isolated successes . . .

- Mostly, we content ourselves with automating “routine” parts of the proof.

- A theorem-prover provides a platform for the deductive ver. approach
  - + expressive, can develop special strategies automating common proof patterns
  - + automatically check proof after changing specs
  - + successful in large critical systems, e.g., NASA, Transportation systems
  - - not automatic in general
  - - requires more expertise (human guidance) than model-checking
Automating the routine

- We can automate linear inequality reasoning:

\[ a \leq x \land b \leq y \land |x - y| < |x - a| \land |x - y| < |x - b| \land \\
(b \leq x \Rightarrow |x - a| < |x - b|) \land (a \leq y \Rightarrow |y - b| < |x - a|) \Rightarrow a = b \]

and basic algebraic re-arrangement:

\[
(w_1^2 + x_1^2 + y_1^2 + z_1^2) \cdot (w_2^2 + x_2^2 + y_2^2 + z_2^2) = \\
(w_1 \cdot w_2 - x_1 \cdot x_2 - y_1 \cdot y_2 - z_1 \cdot z_2)^2 + \\
(w_1 \cdot x_2 + x_1 \cdot w_2 + y_1 \cdot z_2 - z_1 \cdot y_2)^2 + \\
(w_1 \cdot y_2 - x_1 \cdot z_2 + y_1 \cdot w_2 + z_1 \cdot x_2)^2 + \\
(w_1 \cdot z_2 + x_1 \cdot y_2 - y_1 \cdot x_2 + z_1 \cdot w_2)^2
\]
Can also automate some purely logical reasoning such as this:

\[
(\forall x \, y \, z. \, P(x, y) \land P(y, z) \Rightarrow P(x, z)) \land \\
(\forall x \, y \, z. \, Q(x, y) \land Q(y, z) \Rightarrow Q(x, z)) \land \\
(\forall x \, y. \, Q(x, y) \Rightarrow Q(y, x)) \land \\
(\forall x \, y. \, P(x, y) \lor Q(x, y)) \Rightarrow (\forall x \, y. \, P(x, y)) \lor (\forall x \, y. \, Q(x, y))
\]

This is not obvious for most people.
The 17 provers of the world

- Freek Wiedijk’s book *The Seventeen Provers of the World* (Springer-Verlag, Lecture Notes in Computer Science volume 3600) describes:
  - HOL, Mizar, **PVS**, Coq, Otter/IVY, Isabelle/Isar, Alfa/Agda, ACL2,
  - PhoX, IMPS, Metamath, Theorema, Lego, Nuprl, Omega, B prover,
  - Minlog.

- Each one has a proof that $\sqrt{2}$ is irrational 😊
- There are many other systems besides these . . .
Benefits of Theorem-proving & The Pain

- Richer formalisms can express properties that are not, even in principle, solvable by fully automated methods.

- Formalize and verify properties including the underlying theory and assumptions, rather than isolated properties.

- It can be more intellectually stimulating since in the proof process one understands the design more deeply.

- The Pain
  - Even the best interactive theorem provers are difficult to use.
  - For most verifications, more difficult and time-consuming than highly automated methods like model-checking.
Challenges in RTS deductive verification

- Building models
  - Non-trivial problem: models should represent faithfully the application software, and the underlying platform abstraction.
  - Models represent the dynamics of the interaction w.r.t. actions and time.

- RTS deductive verification needs theories of real numbers implemented in a theorem-prover (for RTS verification)
  - Properties of reals are different from properties of naturals or integers.

- Verification is very costly: we currently see use of theorem proving where:
  - The cost of error is too high, e.g. $475M for the floating-point bug in the Intel Pentium processor.
  - Safety-critical niche
Overview of the remainder of the lecture

• Deductive Logic

• Basic notions of
  • Propositional Logic
  • Predicate/First-order Logic (FOL)
  • Higher-order Logic (HOL)

• Proof Formats
  • Structured Derivations
  • Sequent Calculus

• Proving Invariance Properties on Transition Systems
Deductive Logic

• Deductive reasoning or deductive logic: reasoning that uses deductive arguments to move from given statements (premises) to conclusions, which must be true if the premises are true.

• Example of deductive reasoning, given by Aristotle, is

  All men are mortal. (major premise)
  Socrates is a man. (minor premise)
  Socrates is mortal. (conclusion)

• An argument is deductively valid if, whenever all premises are true, the conclusion is also necessarily true.
Sound Arguments

- An argument is **sound** if and only if
  - The argument is valid.
  - All of its premises are true.

**Example 1:**
- All men are mortal.
- Socrates is a man.
- Therefore, Socrates is mortal.

**Sound:** The argument is valid (conclusion follows from the premises) and the premises are in fact true.

**Example 2:**
- All organisms with wings can fly.
- Penguins have wings.
- Therefore, penguins can fly.

**Not sound:** Since the first premise is actually false, the argument, though valid, is not sound.
# Propositional Logic

<table>
<thead>
<tr>
<th>English</th>
<th>Formal</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>false</em></td>
<td>$\perp$</td>
</tr>
<tr>
<td><em>true</em></td>
<td>$T$</td>
</tr>
<tr>
<td>not $P$</td>
<td>$\neg P$</td>
</tr>
<tr>
<td>$P_1$ and $P_2$</td>
<td>$P_1 \land P_2$</td>
</tr>
<tr>
<td>$P_1$ or $P_2$</td>
<td>$P_1 \lor P_2$</td>
</tr>
<tr>
<td>$P_1$ implies $P_2$</td>
<td>$P_1 \Rightarrow P_2$</td>
</tr>
<tr>
<td>$P_1$ iff $P_2$</td>
<td>$P_1 \iff P_2$</td>
</tr>
</tbody>
</table>
Propositional Logic (cont’d)

- Precedence order of logical operators:
  \[ \neg, \land, \lor, \Rightarrow, \Leftrightarrow \]

- Example: \((p \land q) \Rightarrow r\)
  \[\Leftrightarrow\]
  \[p \land q \Rightarrow r\]
Propositional Logic (cont’d)

- **Proposition Equivalence:**
  - $X, Y$ equivalent iff $X = Y$ is tautology (true in every state!)

- **Commutative Laws** ($*: \{\land, \lor, =\}$)
  - $X * Y = Y * X$

- **Distributive Laws:**
  - $X \lor (Y \land Z) = (X \lor Y) \land (X \lor Z)$
  - $X \land (Y \lor Z) = (X \land Y) \lor (X \land Z)$

- **De Morgan Laws:**
  - $\neg (X \land Y) = \neg X \lor \neg Y$
  - $\neg (X \lor Y) = \neg X \land \neg Y$

- **Absorption Law:**
  - $X \land (X \lor Y) = X$

- The rest on pages 20 – 21 in the course material.
First-order Logic (Predicate Logic)

- Predicate: function $P: T \rightarrow \{\text{false, true}\}$
  - predicate on $T$ (type of the predicate). When $P$ is a predicate on $T$, we sometimes say $P$ is a property of $T$.

- FOL covers predicates and quantification over propositions.

Example: $p = \"Socrates is a man\"$, $q = \"Plato is a man\"$.
  - In FOL, $p$ and $q$ can be both expressed by predicate $\text{Man}(x)$, where $\text{Man}(x)$ means that $x$ is a man.
  - When $x = \text{Socrates}$ we get the first proposition, $p$, and when $x = \text{Plato}$ we get the second proposition, $q$.

- Much more powerful logic:
  - universal quantifiers (for all, $\forall$):
    Example: "for every $x$, if $\text{Man}(x)$, then...".
  - existential quantifiers (there exists, $\exists$):
    Example: $\exists x . (\text{Man}(x) \land (\forall y . (\text{time}(y) \Rightarrow \text{CanBeFooled}(x, y))))$
<table>
<thead>
<tr>
<th>English</th>
<th>Formal</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>false</em></td>
<td>( \bot )</td>
</tr>
<tr>
<td><em>true</em></td>
<td>( \top )</td>
</tr>
<tr>
<td>not ( P )</td>
<td>( \neg P )</td>
</tr>
<tr>
<td>( P_1 ) and ( P_2 )</td>
<td>( P_1 \land P_2 )</td>
</tr>
<tr>
<td>( P_1 ) or ( P_2 )</td>
<td>( P_1 \lor P_2 )</td>
</tr>
<tr>
<td>( P_1 ) implies ( P_2 )</td>
<td>( P_1 \Rightarrow P_2 )</td>
</tr>
<tr>
<td>( P_1 ) iff ( P_2 )</td>
<td>( P_1 \Leftrightarrow P_2 )</td>
</tr>
<tr>
<td>for all ( x ), ( P )</td>
<td>( \forall x. \ P(x) )</td>
</tr>
<tr>
<td>there exists ( x ) such that ( P )</td>
<td>( \exists x. \ P(x) )</td>
</tr>
</tbody>
</table>
Higher-order Logic (HOL)

- Allows quantifications over functions.
- In FOL, it is forbidden to quantify over predicates.

- A higher-order predicate is a predicate that takes one or more other predicates as arguments.

- In general, a higher-order predicate of order $n$ takes one or more $(n - 1)$th-order predicates as arguments, where $n > 1$. A similar remark holds for higher-order functions.

+ HOL is a more expressive logic than FOL
- HOL is less analyzable (Gödel: there is no sound and complete proof calculus for HOL)
Proof Rules of Deductive Verification

- A deductive system contains a set of well-defined proof rules called inference rules.

- The system of formal proof rules allows certain formulae to be established as “theorems”.

- Proof rule: \[
\phi_1 \cdot \cdot \cdot \phi_n \quad \psi \\
\phi_1 \\
\phi_n
\]

- If premises \(\phi_1 \land \ldots \land \phi_n\) are valid, then so is the conclusion \(\psi\).

- The premises are assertions (non-temporal formulas).
- The conclusion can be temporal (but not necessarily).
Validity

- $\models p$: **General** Validity — Formula $p$ is valid over all models.

- $\Gamma \models p$: **$\Gamma$-state** validity — Assertion $p$ is valid over all $\Gamma$-reachable states.

- $\Gamma |\models p$: **$\Gamma$-validity** — Formula $p$ is valid over all $\Gamma$-computations.
<table>
<thead>
<tr>
<th>Introduction</th>
<th>Elimination</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\land$</td>
<td>$\land i$</td>
</tr>
<tr>
<td>$\lor$</td>
<td>$\lor i_1$, $\lor i_2$</td>
</tr>
<tr>
<td>$\rightarrow$</td>
<td>$\phi \rightarrow \psi \rightarrow c$</td>
</tr>
<tr>
<td>$\leftrightarrow$</td>
<td>$\phi \leftrightarrow \psi \leftrightarrow \phi \leftrightarrow i$</td>
</tr>
<tr>
<td>$\neg$</td>
<td>$\neg \phi \neg \phi \neg c$</td>
</tr>
</tbody>
</table>

Should be applied while developing a proof!
Additonal Proof Rules

\[
\frac{\phi \rightarrow \psi}{\neg \phi \lor \psi} \quad (\rightarrow 2 \lor) \quad \frac{\neg \phi \lor \psi}{\phi \rightarrow \psi} \quad (\lor 2 \rightarrow)
\]

\[
\frac{\phi \rightarrow \psi \quad \neg \psi}{\neg \phi} \quad \text{MT}
\]

\[
\frac{\neg \phi}{\bot} \quad \text{RAA}
\]

\[
\frac{\phi \lor \neg \phi}{\phi} \quad \text{LEM}
\]

MT : Modus Tollens

RAA: Reductio ad absurdum

LEM: Law of the Excluded Middle

Identity rules: \( \models \phi \land T \leftrightarrow \phi \), \( \models \phi \lor \bot \leftrightarrow \phi \)

Dominance rules: \( \models \phi \lor T \leftrightarrow T \), \( \models \phi \land \bot \leftrightarrow \bot \)
Proof Formats

- **Structured derivation**: a calculational paradigm for manipulating mathematical expressions (attributed to W. Feijen)
  - The initial expression is transformed step by step
  - Each new version of the expression is written on a new line
  - Between the two lines is written a symbol denoting the relationship between the expressions
  - Between the two lines is also written a justification for the step validity

- Example: \( 2x = 2(x + 1) - 2 \)
  \( \iff \{\text{simplify right-hand side}\} \) (hint why)
  \( 2x = 2x \)
  \( \iff \{\text{equality is reflexive}\} \) (hint why)
  \( T \)
Example of Proof as Structured Derivation

• Distribution of $\land$ over $\lor$

**Theorem:** For any $X$, $Y$, $Z$:

$$X \land (Y \lor Z) = (X \land Y) \lor (X \land Z)$$

**Proof:**

$$(X \land Y) \lor (X \land Z)$$

$= \quad \{ \lor \text{ distributes over } \land \}$$

$$((X \land Y) \lor X) \land ((X \land Y) \lor Z)$$

$= \quad \{ \text{Law of Absorption} \}$$

$$X \land ((X \land Y) \lor Z)$$

$= \quad \{ \lor \text{ distributes over } \land \}$$

$$X \land (X \lor Z) \land (Y \lor Z)$$

$= \quad \{ \text{Law of Absorption} \}$$

$$X \land (Y \lor Z)$$
Proof Formats (cont’d)

- **Sequent**: a proof goal is a sequent (a sequence of formulas)
  - a sequent $S$ is represented as:
    $$
    \{1\} A_1 \\
    \{2\} A_2 \\
    \{3\} A_3 ... \\
    \|--- --- ---
    \{1\} B_1
    \{2\} B_2
    \{3\} B_3 ... 
    $$
  - $A_1, A_2, A_3 ...$ are called antecedents; $B_1, B_2, B_3 ...$ are consequents.
  - Interpretation: $A_1 \land A_2 \land A_3 \land ... \Rightarrow B_1 \lor B_2 \lor B_3 \lor ...$
Proofs in Sequent Calculus

Proofs are done by transforming the sequent until one of the following forms is obtained:

$$\begin{array}{c}
\vdots \\
\phi \\
\vdots \\
\phi \\
\hline
\end{array}$$

i.e. $\Gamma, \phi \vdash \phi \lor \ldots$

which is a case of Rule Premise and $\lor \iota$

$$\begin{array}{c}
\vdots \\
\hline
\vdots \\
\end{array}$$

i.e. $\Gamma \vdash T \lor \ldots$

which is a case of Dominance of $T$

$$\begin{array}{c}
\vdots \\
\hline
\bot \\
\vdots \\
\end{array}$$

i.e. $\Gamma, \bot \vdash \ldots$

Which is a case of $\bot \iota$. 
Check Validity of Arguments

• Check if the sequent is a valid argument – prove the theorem:

\[ \text{V1: THEOREM } A_1 \land \ldots \land A_n \Rightarrow B_1 \lor \ldots \lor B_n \]

• Check consistency of premises – prove the theorem:

\[ \text{V2: THEOREM } A_1 \land \ldots \land A_n \Rightarrow \text{FALSE} \]

\[ \Leftarrow \{ \text{ why? } \} \]

\[ \text{V2: THEOREM } \neg ( A_1 \land \ldots \land A_n ) \]
**Deductive Invariance Proofs**

- **Safety property**: A property that can be specified by \([\Box p \text{ (LTL)}]\)

- The properties above: invariance properties (of model \(\Sigma\))

- Given a transition system (discrete or timed), we denote:
  
  - \(V\) – A finite set of typed state variables. A \(V\)-state \(s\) is an interpretation of \(V\). 
    \(\Sigma_V\) – the set of all \(V\)-states.
  
  - \(\emptyset \subseteq V\) – A set of observable variables.
  
  - \(\Theta\) – An initial condition. A satisfiable assertion that characterizes the initial states.
  
  - \(\rho\) – A transition relation. An assertion \(\rho(V,V')\), referring to both unprimed (current) and primed (next) versions of the state variables. For example, \(x' = x + 1\) corresponds to the assignment \(x := x + 1\).
Basic Invariance Proofs

• The simplest rule for proving invariance properties: Basic INV
  • establishes p as an invariant of some model P.

Rule Basic INV

I1. \( \Theta \rightarrow p \)
I2. \( p \land \rho_{\tau} \rightarrow p' \quad (\{p\} \tau \{p\}) \)

\[ \therefore \quad [] p \]

An assertion/predicate p satisfying I1 and I2 is called inductive.

• Premise I1 (proof obligation/verification condition): requires the initial condition \( \Theta \) to imply property p.

• Premise I2 (proof obligation/verification condition) : requires that all transitions in P preserve p.