An Informal Introduction to *Formal Methods*

by
C. Michael Holloway
Paul S. Miner
Ricky W. Butler
Assessment Technology Branch

Structure

- Definitions & Introduction to Basics of Logic
  - -- 30 minutes
  - -- Michael Holloway
- Application of Logic to Digital System Design
  - -- 30 minutes
  - -- Paul Miner
- *Break for 30 minutes*
- Detailed Example
  - -- 1 hour
  - -- Ricky Butler

Purposes

- The five main purposes of these presentations are to
  - -- Provide a high-level overview of some of the foundations of formal methods
  - -- Demonstrate the variety of formal techniques that are available
  - -- Illustrate ways in which formal methods may be applied in various domains
  - -- Set the context for the remaining presentations at this workshop
  - -- Motivate future in-depth study of formal methods

Purposes

- The five main purposes of these presentations are to
  - Provide a high-level overview of some of the foundations of formal methods
  - Demonstrate the variety of formal techniques that are available
  - Illustrate ways in which formal methods may be applied in various domains
  - Set the context for the remaining presentations at this workshop
  - Motivate future in-depth study of formal methods

Non-Purposes

- These presentations are not intended to
  - Enable you to carry on a meaningful conversation with mathematical logicians
  - Present a balanced view of the relative capabilities of existing formal methods tools and techniques
  - Discuss all of the domains in which formal methods may be applied
  - Tell you all you need to know to become a formal methods expert
  - Put you to sleep

What Does "Formal" Mean?

When many people think of the word "formal", they think of something like this:

What Does "Formal" Mean?

When many people think of the word "formal", they think of something like this:

What Does "Formal" Mean?
(continued)

- Others think "formal" means one or more of the following things:
  - boring, uninteresting
  - difficult to understand, complicated
  - filled with many odd symbols and foreign letters

$\forall \quad \exists \quad \supset \quad \lambda \quad \emptyset \quad \wedge \quad \vee \quad \aleph \quad \in \quad \equiv \quad \psi \quad \varphi$

- One of the goals of this workshop is to convince you that none of these perceptions is necessarily true

What Does “Formal” Mean?

When many people think of the word “formal”, they think of something like this:

What Does “Formal” Mean?

(continued)

- Others think “formal” means one or more of the following things:
  - -- boring, uninteresting
  - -- difficult to understand, complicated
  - -- filled with many odd symbols and foreign letters

    $\forall \quad \exists \quad \supset \quad \lambda \quad \varnothing \quad \wedge \quad \vee \quad \aleph \quad \in \quad \equiv \quad \psi \quad \varphi$

- One of the goals of this workshop is to convince you that none of these perceptions is necessarily true

What Does “Formal” Mean?

(continued)

- Webster’s dictionary gives the following as one of the definitions of “formal”:

  relating to, concerned with, or constituting the outward form of something as distinguished from its content

- A method is formal if its rules for manipulation are based on form (syntax) and not on content (semantics)
- Simple arithmetic is an example of such a method:
  - -- 2 cats + 2 cats = 4 cats
  - -- 2 mice + 2 mice = 4 mice
  - -- 2x + 2x = 4x

Validity of addition rests on its form alone --- as long as like objects are added, it doesn’t matter what those objects are.

2 cats + 2 mice = ????

What Does “Formal” Mean?

(continued)

		play volleyball	go hiking			read a book	play baseball	
Temp	hot cool	hot cool	cool	hot	*	hot cool	hot cool	hot
Weather	sunny cloudy rain snow	sunny	sunny cloudy	cloudy	snow	rain	cloudy	sunny
Windy?	true false	false	true	true	*	true false	false	true

Is this table “formal” ?

What Does “Formal” Mean?

(continued)

Parameters

Possible Values

Operational Procedures (possible actions)

		play volleyball	go hiking			read a book	play baseball	
Temp	hot cool	hot cool	cool	hot	*	hot cool	hot cool	hot
Weather	sunny cloudy rain snow	sunny	sunny cloudy	cloudy	snow	rain	cloudy	sunny
Windy?	true false	false	true	true	*	true false	false	true

Example

Scenarios (conditions under which a particular op. proc. is chosen)

If ((Temp = hot) or (Temp = cool)) and (Weather = sunny) and not Windy? then play volleyball

Yes, it is.

Symbolic Logic

- The majority of existing formal techniques are based on some flavor of symbolic logic
- Remainder of my talk will provide brief informal introduction to each of the following:
  - -- Propositional Logic
  - -- Predicate Logic
  - -- Other Logics
- Along the way, we'll also discuss some of the difficulties in translating natural language into a logic

Propositional Logic

- A *proposition* is a statement that is either true or false
  - -- Today is Wednesday
  - -- I have a son named David
  - -- 2 + 2 = 5
  - -- Fermat's last theorem is true
  - -- This workshop is the most valuable workshop I've ever attended
- The possibility of assigning a truth value is all that is required; actual determination of the truth value is not required
- Questions, commands, etc., are not propositions

Propositional Logic

(connectives)

- Propositions may be combined using the following *connectives*:
  - -- **and** ($\wedge$): $P \wedge Q$ is true when both P and Q are true
  - -- **or** ($\vee$): $P \vee Q$ is true when either P or Q is true
  - -- **not** ($\neg$): $\neg P$ is true when P is false
  - -- **if-then** ($\Rightarrow, \supset$): $P \supset Q$ is true except when P is true and Q is false
  - -- **if-and-only-if** ($\equiv$): $P \equiv Q$ is true when P and Q have the same truth value

Propositional Logic
(connectives)

- Propositions may be combined using the following *connectives*:

  -- **and** ($\wedge$): $P \wedge Q$ is true when both P and Q are true

  -- **or** ($\vee$): $P \vee Q$ is true when either P or Q is true

  -- **not** ($\neg$): $\neg P$ is true when P is false

  -- **if-then** ($\Rightarrow, \supset$): $P \supset Q$ is true except when P is true and Q is false

  -- **if-and-only-if** ($\equiv$): $P \equiv Q$ is true when P and Q have the same truth value

Propositional Logic
(connectives vs. natural language)

- Please note that each connective has a precisely defined meaning
- This meaning does not always correspond to the natural language meaning
- The definition of **if-then** especially tends to confuse many people, because it does not require cause and effect

$$\left.\begin{array}{l} 2+2=4 \supset 2\times 2=4 \\ 2+2=5 \supset 2\times 2=4 \\ 2+2=5 \supset 2\times 2=5 \end{array}\right\} \text{are all true}$$

$$2+2=4 \supset 2\times 2=5 \quad \text{is false}$$

Propositional Logic
(connectives)

- Propositions may be combined using the following *connectives*:

  -- **and** ($\wedge$): $P \wedge Q$ is true when both P and Q are true

  -- **or** ($\vee$): $P \vee Q$ is true when either P or Q is true

  -- **not** ($\neg$): $\neg P$ is true when P is false

  -- **if-then** ($\Rightarrow, \supset$): $P \supset Q$ is true except when P is true and Q is false

  -- **if-and-only-if** ($\equiv$): $P \equiv Q$ is true when P and Q have the same truth value

Propositional Logic
(properties)

- Some properties (axioms & theorems) of propositional logic

  -- **commutativity**: $P \wedge Q \equiv Q \wedge P$, $P \vee Q \equiv Q \vee P$

  -- **associativity**: $(P \wedge Q) \wedge R \equiv P \wedge (Q \wedge R)$, same for $\vee$

  -- **distributivity**: $P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R)$
  $P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$

  -- **De Morgan's laws:** $\neg(P \wedge Q) \equiv \neg P \vee \neg Q$
  $\neg(P \vee Q) \equiv \neg P \wedge \neg Q$

  -- **double negation**: $\neg\neg P \equiv P$

  -- **implication definition**: $P \supset Q \equiv \neg P \vee Q$

Propositional Logic
(inference rules)

- *Inference rules* permit reasoning (deducing conclusions based on the truth of certain premises)

- Some inference rules of propositional logic include

modus ponens	*modus tollens*
$P \supset Q$	$P \supset Q$
P	$\neg Q$
Q	$\neg P$

Propositional Logic
(connectives)

- Propositions may be combined using the following *connectives*:

  -- **and** ($\wedge$): $P \wedge Q$ is true when both P and Q are true

  -- **or** ($\vee$): $P \vee Q$ is true when either P or Q is true

  -- **not** ($\neg$): $\neg P$ is true when P is false

  -- **if-then** ($\Rightarrow, \supset$): $P \supset Q$ is true except when P is true and Q is false

  -- **if-and-only-if** ($\equiv$): $P \equiv Q$ is true when P and Q have the same truth value

Propositional Logic
(properties)

- Some properties (axioms & theorems) of propositional logic

  -- **commutativity**: $P \wedge Q \equiv Q \wedge P$, $P \vee Q \equiv Q \vee P$

  -- **associativity**: $(P \wedge Q) \wedge R \equiv P \wedge (Q \wedge R)$, same for $\vee$

  -- **distributivity**: $P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R)$
  $P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$

  -- **De Morgan's laws:** $\neg(P \wedge Q) \equiv \neg P \vee \neg Q$
  $\neg(P \vee Q) \equiv \neg P \wedge \neg Q$

  -- **double negation**: $\neg\neg P \equiv P$

  -- **implication definition**: $P \supset Q \equiv \neg P \vee Q$

Propositional Logic

(proofs)

- A **proof** of a given proposition establishes the truth of the proposition based only on axioms, theorems, and inference rules
- For propositional logic, truth tables provide a means of defining truth
- Propositional logic is
  - -- *complete*: everything that is true may be proven
  - -- *consistent (sound)*: nothing that is false may be proven
  - -- *decidable*: there exists a mechanical procedure for deciding whether any proposition is true or false

Propositional Logic

(proof example)

- Show that $P \supset (Q \supset S) \equiv (P \wedge Q) \supset S$
- Proof

$P \supset (Q \supset S)$	$\equiv \neg P \vee (Q \supset S)$	*by implication definition*
	$\equiv \neg P \vee (\neg Q \vee S)$	*by implication definition*
	$\equiv (\neg P \vee \neg Q) \vee S$	*by associativity*
	$\equiv \neg(P \wedge Q) \vee S$	*by De Morgan*
	$\equiv (P \wedge Q) \supset S$	*by implication definition*
		QED

Predicate Logic

- Propositional logic only permits reasoning about (true or false) complete sentences
- There is no way to reason about individual objects
  - -- There is a number n where $n \times n = n + n$
  - -- All cats have whiskers
- Predicate logic extends the formal language to permit reasoning about objects and the relationships between them
  - -- $\exists n : n \times n = n + n$
  - -- $\forall a : \text{Cat}(a) \supset \text{Whiskers}(a)$

Predicate Logic
(terms & propositions)

- Objects in predicate logic are denoted by expressions called *terms*:
  - -- Constants a, b, c, ... are terms
  - -- Variables u, v, w, ... are terms
  - -- If $t_1, t_2, \ldots, t_n$ are terms and f is a symbol that denotes a function of n arguments, then $f(t_1, t_2, \ldots, t_n)$ is a term
- In predicate logic, propositions are defined as follows:
  - -- *true* and *false* are propositions
  - -- If $t_1, t_2, \ldots, t_n$ are terms and *p* is a symbol that denotes a predicate of n arguments, then $p(t_1, t_2, \ldots, t_n)$ is a proposition

NASA Langley

Predicate Logic
(terms & propositions)

- Objects in predicate logic are denoted by expressions called *terms*:
  - -- Constants a, b, c, ... are terms
  - -- Variables u, v, w, ... are terms
  - -- If $t_1, t_2, \ldots, t_n$ are terms and f is a symbol that denotes a function of n arguments, then $f(t_1, t_2, \ldots, t_n)$ is a term
- In predicate logic, propositions are defined as follows:
  - -- *true* and *false* are propositions
  - -- If $t_1, t_2, \ldots, t_n$ are terms and *p* is a symbol that denotes a predicate of n arguments, then $p(t_1, t_2, \ldots, t_n)$ is a proposition

NASA Langley

Predicate Logic
(sentences & connectives)

- Every proposition is also a *sentence* (or *formula*)
- Sentences may also be constructed using the connectives from propositional logic; if S and T are sentences, so are
  $S \wedge T,\ S \vee T,\ \neg S,\ S \supset T,\ S \equiv T$
  the meaning of each of these is as expected
- If x is a variable and S is a sentence, then the following are also sentences:
  - -- $\forall x{:}\ S$ true if S is true for all possible values of x
  - -- $\exists x{:}\ S$ true if S is true for at least one value of x

NASA Langley

Predicate Logic
(terms & propositions)

- Objects in predicate logic are denoted by expressions called *terms*:
  - -- Constants a, b, c, ... are terms
  - -- Variables u, v, w, ... are terms
  - -- If $t_1, t_2, \ldots, t_n$ are terms and f is a symbol that denotes a function of n arguments, then $f(t_1, t_2, \ldots, t_n)$ is a term
- In predicate logic, propositions are defined as follows:
  - -- *true* and *false* are propositions
  - -- If $t_1, t_2, \ldots, t_n$ are terms and *p* is a symbol that denotes a predicate of n arguments, then $p(t_1, t_2, \ldots, t_n)$ is a proposition

NASA Langley

Predicate Logic
(sentences & connectives)

- Every proposition is also a *sentence* (or *formula*)
- Sentences may also be constructed using the connectives from propositional logic; if S and T are sentences, so are
  $S \wedge T,\ S \vee T,\ \neg S,\ S \supset T,\ S \equiv T$
  the meaning of each of these is as expected
- If x is a variable and S is a sentence, then the following are also sentences:
  - -- $\forall x{:}\ S$ true if S is true for all possible values of x
  - -- $\exists x{:}\ S$ true if S is true for at least one value of x

NASA Langley

Predicate Logic
(properties)

- Some properties of predicate logic

$$\neg\forall x{:}\ S \equiv \exists x{:}\ \neg S \qquad \neg\exists x{:}\ S \equiv \forall x{:}\ \neg S$$
$$\forall x{:}\ S \equiv \neg\exists x{:}\ \neg S \qquad \exists x{:}\ S \equiv \neg\forall x{:}\ \neg S$$
$$\forall x{:}\ \forall y{:}\ S \equiv \forall y{:}\ \forall x{:}\ S \qquad \exists x{:}\ \exists y{:}\ S \equiv \exists y{:}\ \exists x{:}\ S$$

- The following are true if x is not a free variable in S

$$S \vee \forall x{:}\ T \equiv \forall x{:}\ (S \vee T) \qquad S \vee \exists x{:}\ T \equiv \exists x{:}\ (S \vee T)$$
$$S \wedge \forall x{:}\ T \equiv \forall x{:}\ (S \wedge T) \qquad S \wedge \exists x{:}\ T \equiv \exists x{:}\ (S \wedge T)$$

NASA Langley

Predicate Logic
(translating from English)

- Translating natural language sentences into predicate logic is sometimes fairly simple
- Consider the following sentence:

All humans are mortal

- Using the predicates human(x) and mortal(x), we have

$$\forall x\text{: human}(x) \supset \text{mortal}(x)$$

Predicate Logic
(translating from English incorrectly)

- Translating natural language sentences into predicate logic is sometimes a little tricky
- Consider the following sentence:

Apples and oranges are delicious and nutritious

- Using the predicates apple(x), orange(x), delicious(x), and nutritious(x), we might try the following:

$$\forall x\text{: (apple}(x) \wedge \text{orange}(x)) \supset (\text{delicious}(x) \wedge \text{nutritious}(x))$$

- Is this what we want?

Predicate Logic
(translating from English correctly)

- This sentence does not have the intended meaning: it requires that an object be both an apple and an orange
- The following is what we really want:

$$\forall x\text{: (apple}(x) \vee \text{orange}(x)) \supset (\text{delicious}(x) \wedge \text{nutritious}(x))$$

- Here are a few more sentences for you to try at your leisure
  - -- Government workers are not always lazy
  - -- Testing never reveals the absence of errors
  - -- Only early registrants can attend the dinner at Captain George's
  - -- Nothing of importance was said at the workshop
  - -- A company creates good software if and only if it uses formal methods

Predicate Logic
(translating from English incorrectly)

- Translating natural language sentences into predicate logic is sometimes a little tricky
- Consider the following sentence:

Apples and oranges are delicious and nutritious

- Using the predicates apple(x), orange(x), delicious(x), and nutritious(x), we might try the following:

$$\forall x\text{: (apple}(x) \wedge \text{orange}(x)) \supset (\text{delicious}(x) \wedge \text{nutritious}(x))$$

- Is this what we want?

Predicate Logic

(translating from English correctly)

- This sentence does not have the intended meaning: it requires that an object be both an apple and an orange
- The following is what we really want:

$$\forall x\text{: } (\text{apple}(x) \vee \text{orange}(x)) \supset (\text{delicious}(x) \wedge \text{nutritious}(x))$$

- Here are a few more sentences for you to try at your leisure
  - Government workers are not always lazy
  - Testing never reveals the absence of errors
  - Only early registrants can attend the dinner at Captain George's
  - Nothing of importance was said at the workshop
  - A company creates good software if and only if it uses formal methods

Predicate Logic

(translating from English: answers)

Government workers are not always lazy

$$\exists x\text{: government-worker}(x) \wedge \neg\text{lazy}(x)$$

Testing never reveals the absence of errors

$$\forall x\text{: testing}(x) \supset \neg\text{reveal-absence}(x)$$

Only early registrants can attend the dinner at Captain George's

$$\forall x\text{: dinner}(x) \supset \text{early-registrant}(x)$$

Nothing of importance was said at the workshop

$$\forall x\text{: important}(x) \supset \neg\text{said-at-workshop}(x)$$

A company creates good software if and only if it is uses FM

$$\forall x\text{: company}(x) \supset (\text{good-software}(x) \equiv \text{uses-formal-methods}(x))$$

Predicate Logic

(proofs)

- A key concept in developing proofs in predicate logic is *skolemization*
  - Provides procedure for reducing a sentence into a propositional sentence that is true if the original sentence is true
  - Based on appropriate elimination of quantifiers
- Predicate logic is *complete* and *consistent,* but is only *semidecidable:*
  - There is a mechanical procedure that will terminate for all true sentences
  - This procedure may not terminate for all false sentences, but it will identify the sentence as false if it does terminate

Predicate Logic

(proof example)

- Consider the sentence

$$\forall x\text{: }\exists y\text{: } y = x + 7$$

- We first replace the universally quantified variable x by the arbitrary constant a

$$\exists y\text{: } y = a + 7$$

- We may now remove the existential quantifier, giving the Skolem form for the sentence

$$y = a + 7$$

- This is easily satisfied by letting $y = a + 7$

$$a + 7 = a + 7$$

QED

Predicate Logic

(proofs)

- A key concept in developing proofs in predicate logic is *skolemization*
  - Provides procedure for reducing a sentence into a propositional sentence that is true if the original sentence is true
  - Based on appropriate elimination of quantifiers
- Predicate logic is *complete* and *consistent,* but is only *semidecidable:*
  - There is a mechanical procedure that will terminate for all true sentences
  - This procedure may not terminate for all false sentences, but it will identify the sentence as false if it does terminate

Predicate Logic

(proofs)

- A key concept in developing proofs in predicate logic is *skolemization*
  - -- Provides procedure for reducing a sentence into a propositional sentence that is true if the original sentence is true
  - -- Based on appropriate elimination of quantifiers
- Predicate logic is *complete* and *consistent,* but is only *semidecidable:*
  - -- There is a mechanical procedure that will terminate for all true sentences
  - -- This procedure may not terminate for all false sentences, but it will identify the sentence as false if it does terminate

Other Logics

- *Many-sorted logic* extends predicate logic by dividing the universe from which variables and constants are chosen into several sorts (e.g., integers, reals, characters)
- *Higher order logic* (or *type theory*) allows functions to take functions as arguments and to return functions as values, and permits quantification over functions and predicates
- *Constructive logic* requires that a proof of existence of an object provides a procedure for constructing the object
- *Programming logics* provide rules for specifying and reasoning about imperative programs and program state
- *Modal logics*, *temporal logics*, *logics for partial functions*

Other Logics

- *Many-sorted logic* extends predicate logic by dividing the universe from which variables and constants are chosen into several sorts (e.g., integers, reals, characters)
- *Higher order logic* (or *type theory*) allows functions to take functions as arguments and to return functions as values, and permits quantification over functions and predicates
- *Constructive logic* requires that a proof of existence of an object provides a procedure for constructing the object
- *Programming logics* provide rules for specifying and reasoning about imperative programs and program state
- *Modal logics*, *temporal logics*, *logics for partial functions*

Higher Order Logic

(example)

- Consider the following statement:
  - -- There does not exist a list of all the possible functions that take a natural number as an argument and return a natural number as the result
- In a higher order logic, we might express this as follows:

$$
\begin{aligned}
&\neg\exists(\mathrm{list} : \text{function}[\text{nat} \rightarrow \text{function}[\text{nat} \rightarrow \text{nat}]]) \\
&\quad \forall(g : \text{function}[\text{nat} \rightarrow \text{nat}]) \\
&\qquad \exists(i : \text{nat}) \\
&\qquad\quad \mathrm{list}(i) = g
\end{aligned}
$$

Other Logics

- *Many-sorted logic* extends predicate logic by dividing the universe from which variables and constants are chosen into several sorts (e.g., integers, reals, characters)
- *Higher order logic* (or *type theory*) allows functions to take functions as arguments and to return functions as values, and permits quantification over functions and predicates
- *Constructive logic* requires that a proof of existence of an object provides a procedure for constructing the object
- *Programming logics* provide rules for specifying and reasoning about imperative programs and program state
- *Modal logics*, *temporal logics*, *logics for partial functions*

Higher Order Logic

(example)

- Consider the following statement:
  - There does not exist a list of all the possible functions that take a natural number as an argument and return a natural number as the result
- In a higher order logic, we might express this as follows:

$$\neg\exists(\text{list} : \text{function}[\text{nat} \rightarrow \text{function}[\text{nat} \rightarrow \text{nat}]])$$
$$\forall(g : \text{function}[\text{nat} \rightarrow \text{nat}])$$
$$\exists(i : \text{nat})$$
$$\text{list}(i) = g$$

Higher Order Logic

(example, continued)

- What does

$$\text{list}(i) = g$$

  mean?
- Two functions are defined to be equal if for every possible argument value, they return the same result
- This can be expressed in our notation as

$$\forall(x : \text{nat})$$
$$\text{list}(i)(x) = g(x)$$

Concluding Remarks

- Of the five main purposes mentioned at the beginning, I hope that my presentation has
  - Provided a high-level overview of some of the foundations of formal methods
  - Begun to set the context for the remaining presentations at this workshop
  - Perhaps motivated future in-depth study of formal methods
- The remaining presentations should continue to motivate and to set context, and should also
  - Demonstrate the variety of formal techniques that are available
  - Illustrate ways in which formal methods may be applied in various domains