Practical Reasoning About the Real World using Logic and Computation

Rajeev Goré

Logic and Computation Group
Research School of Computer Science
The Australian National University
rajeev.gore@anu.edu.au

March 4, 2015
Overview: Practical Reasoning About the Real World

Problems from the Real World
Solutions Using Logic and Computation
What is involved in L&C research
Connections with Other Research Areas
How you can join us
Demo: modelling the days of the week using logic
Knowledge Base

KB = \[
\begin{align*}
A \land \text{okX} & \rightarrow \neg B \\
\neg A \land \text{okX} & \rightarrow B \\
B \land \text{okY} & \rightarrow \neg C \\
\neg B \land \text{okY} & \rightarrow C
\end{align*}
\]

Is A, \neg C normal behavior?
Reasoning about Security Protocols

Alice (A) and Bob (B) use a trusted server (S) to distribute public keys on request

1. A requests B’s public key $K_{PB}$ from S
2. S responds with B’s public key $K_{PB}$ alongside B’s identity, signed by the server using $K_{SS}$ for authentication
3. A invents $N_A$ and sends it to B using B’s public key $K_{PB}$
4. B requests A’s public key $K_{PA}$ from S
5. S responds with A’s public key $K_{PA}$ alongside A’s identity, signed by the server using $K_{SS}$ for authentication
6. B invents $N_B$, and sends it to A along with $N_A$ to prove ability to decrypt with $K_{SB}$
7. A confirms $N_B$ to B, to prove ability to decrypt with $K_{SA}$

Claim: At end of protocol, A and B know each other’s identities, and know both $N_A$ and $N_B$ but eavesdroppers don’t
Reasoning about Business Processes

Banking: rules for meeting anti-fraud legislation

Example: rules for granting a request to open a bank account

Rule 1: A risk assessment must eventually be carried out for each request to open a bank account
Rule 2: A request to open a bank account is granted only if the risk is assessed as low
Rule 3: A due diligence assessment must eventually be carried out for each request to open a bank account
Rule 4: If a person fails due diligence then he or she must be blacklisted

English: prose used so that bank staff can understand rules
Reasoning about Business Processes

Banking: rules for meeting anti-fraud legislation

Example: rules for granting a request to open a bank account

Rule 1: A risk assessment must eventually be carried out for each request to open a bank account

Rule 2: A request to open a bank account is granted only if the risk is assessed as low

Rule 3: A due diligence assessment must eventually be carried out for each request to open a bank account

Rule 4: If a person fails due diligence then he or she must be blacklisted

English: prose used so that bank staff can understand rules

Question: how to check these rules for consistency and sanity?
Reasoning about Web Applications

Suppose we want to monitor web-based applications.

How to specify things like:

Rule 1: allow a program to connect to a remote site if and only if it has neither tried to open a local file that it has not created, nor tried to modify a file it has created, nor tried to create a sub-process:

Rule 2: allow a program to open local files in user-specified directories for modifications if and only if it has created them, and it has neither tried to connect to a remote site nor tried to create a sub-process.
SNOMED CT (diagnosis support and data exchange)

- caused by induces a relation between medical terms
- is a induces subset relationship
- industry standard in many countries (Australia, UK, ...)

Image courtesy of International Health Terminology Standards Development Organisation
SNOMED CT (diagnosis support and data exchange)

- caused_by induces a relation between medical terms
- is_a induces subset relationship
- industry standard in many countries (Australia, UK, ...)

Question: can we use this for diagnosis
Suppose we want to go beyond standard database query languages and invent better ones.

**Expressivity:** what queries can (cannot) be expressed in SQL?

**Complexity Theory:**
- can we differentiate database size from query size
- can we reason about the combined complexity

**Practicality:**
- Query evaluation $\rightsquigarrow$ query processing
- Query Containment/equivalence $\rightsquigarrow$ query Optimisation
Classical Propositional Logic (CPL)

Syntax: captures “not”, “and”, “or”, “if then”, “iff” via

\[
\begin{align*}
\text{atom} & ::= p \mid q \mid r \mid \cdots \\
\phi, \psi & ::= \text{atom} \mid \neg \phi \mid \phi \land \psi \mid \phi \lor \psi \mid \phi \to \psi \mid \phi \leftrightarrow \psi
\end{align*}
\]
Classical Propositional Logic (CPL)

Syntax: captures “not”, “and”, “or”, “if then”, “iff” via

\[
atom ::= p \mid q \mid r \mid \cdots \\
\varphi, \psi ::= atom \mid \neg \varphi \mid \varphi \land \psi \mid \varphi \lor \psi \mid \varphi \rightarrow \psi \mid \varphi \leftrightarrow \psi
\]

Model: for every atom, an assignment to \( t \) or else to \( f \)

Example Model: \( p \) stands for “I have a brother”
Classical Propositional Logic (CPL)

Syntax: captures “not”, “and”, “or”, “if then”, “iff” via

\[
\text{atom} ::= p \mid q \mid r \mid \cdots \\
\varphi, \psi ::= \text{atom} \mid \neg \varphi \mid \varphi \land \psi \mid \varphi \lor \psi \mid \varphi \rightarrow \psi \mid \varphi \leftrightarrow \psi
\]

Model: for every atom, an assignment to \(t\) or else to \(f\)
Example Model: \(p\) stands for “I have a brother”

Example of truth value of larger formulae: \((p \land (p \rightarrow q)) \rightarrow q\)

A1: \(p := t\) and \(q := t\)

\[
(t \rightarrow t) \text{ evaluates to } t \\
(t \land t) \text{ evaluates to } t \\
(t \rightarrow t) \text{ evaluates to } t
\]

So: \((p \land (p \rightarrow q)) \rightarrow q\) is \(t\) in model \(p := t\) and \(q := t\)
SATisfiability, Validity and Automated Reasoning

Semantics: two important inter-related notions
  Satisfiable: true under some assignment
  Valid: true under all assignments
Lemma: $\varphi$ is valid if and only if $\neg\varphi$ is unsatisfiable
SATisfiability, Validity and Automated Reasoning

Semantics: two important inter-related notions

Satisfiable: true under some assignment
Valid: true under all assignments

Lemma: $\varphi$ is valid if and only if $\neg\varphi$ is unsatisfiable

Example: $(p \land (p \rightarrow q)) \rightarrow q$ is satisfiable  $(p := t$ and $q := t)$

Is it valid?: check all $2^n$ assignments for $n$ atoms  $(n = 2)$

Question: is there a purely syntactic way?
SATisfiability, Validity and Automated Reasoning

Semantics: two important inter-related notions
Satisfiable: true under some assignment
Valid: true under all assignments
Lemma: \( \varphi \) is valid if and only if \( \neg \varphi \) is unsatisfiable

Example: \((p \land (p \rightarrow q)) \rightarrow q\) is satisfiable \((p := t \text{ and } q := t)\)
Is it valid?: check all \(2^n\) assignments for \(n\) atoms \((n = 2)\)
Question: is there a purely syntactic way?

Automated Reasoning: decide satisfiability or validity efficiently on a computer using purely syntactic methods
Goal: input \(\varphi\) and get answer “satisfiable” or “unsatisfiable”
Ideally: output an/all assignment(s) that makes it satisfiable
Ideally: output proof of why it is unsatisfiable
Reasoning about Digital Circuits

Knowledge Base

\[ KB = \{ \begin{align*}
A \land \text{okX} & \rightarrow \neg B \\
\neg A \land \text{okX} & \rightarrow B \\
B \land \text{okY} & \rightarrow \neg C \\
\neg B \land \text{okY} & \rightarrow C
\end{align*} \] 

Is A, \neg C normal behavior?

“normal” ... all gates are okay
Answering Single Query

\begin{align*}
A \land okX & \rightarrow \neg B \\
\neg A \land okX & \rightarrow B \\
B \land okY & \rightarrow \neg C \\
\neg B \land okY & \rightarrow C
\end{align*}

Is A, \neg C normal behavior?

\begin{align*}
A \land okX & \rightarrow \neg B \\
\neg A \land okX & \rightarrow B \\
B \land okY & \rightarrow \neg C \\
\neg B \land okY & \rightarrow C \\
A, \neg C, okX, okY & \rightarrow\rightarrow\rightarrow\rightarrow\rightarrow
\end{align*}

Satisfiability Algorithm
Multiple Queries

A ∧ okX → ¬B
¬A ∧ okX → B
B ∧ okY → ¬C
¬B ∧ okY → C
Banking: rules for meeting anti-fraud legislation

Example: rules for granting a request to open a bank account

Rule 1: A risk assessment (ra) must eventually be carried out for each request to open a bank account (ro)

Rule 2: A request to open a bank account (ro) is granted (rog) only if the risk is assessed as low (ral)

Rule 3: A due diligence assessment (dd) must eventually be carried out for each request to open a bank account (ro)

Rule 4: If a person fails due diligence (ddf) then he or she must be blacklisted (bl)

Question: how to check these rules for consistency and sanity?
Reasoning About Business Processes

Banking: rules for meeting anti-fraud legislation

Example: rules for granting a request to open a bank account

Rule 1: A risk assessment (ra) must eventually be carried out for each request to open a bank account (ro)

Rule 2: A request to open a bank account (ro) is granted (rog) only if the risk is assessed as low (ral)

Rule 3: A due diligence assessment (dd) must eventually be carried out for each request to open a bank account (ro)

Rule 4: If a person fails due diligence (ddf) then he or she must be blacklisted (bl)

Question: how to check these rules for consistency and sanity?

Need: a logic which captures temporal notions like “eventually”, “after”, “before” as well as “if then”, “only if” etc.
Linear Temporal Logic (LTL)

**Syntax:** represents linguistic notions using logical connectives

- **CPL:** captures “and”, “or”, “not”, “if then”, “iff”
- **LTL:** adds “next”, “until”, “before”, “always”, “eventually”

\[
\begin{align*}
\text{atom} & \ ::= \ p \mid q \mid r \mid \cdots \\
\varphi, \psi & \ ::= \ \text{atom} \mid \neg \varphi \mid \varphi \land \psi \mid \varphi \lor \psi \mid \varphi \rightarrow \psi \mid \varphi \leftrightarrow \psi \\
& \quad \mid X\varphi \mid \varphi U \psi \mid \varphi B \psi \mid G \varphi \mid F \varphi
\end{align*}
\]
Linear Temporal Logic (LTL)

Syntax: represents linguistic notions using logical connectives

CPL: captures “and”, “or”, “not”, “if then”, “iff”

LTL: adds “next”, “until”, “before”, “always”, “eventually”

\[
\begin{align*}
\text{atom} & \ ::= \ p \ | \ q \ | \ r \ | \cdots \\
\varphi, \psi & \ ::= \ \text{atom} \ | \ \neg \varphi \ | \ \varphi \land \psi \ | \ \varphi \lor \psi \ | \ \varphi \rightarrow \psi \ | \ \varphi \leftrightarrow \psi \\
& \quad \ | \ X\varphi \ | \ \varphi \ U \psi \ | \ \varphi \ B \psi \ | \ G\varphi \ | \ F\varphi
\end{align*}
\]

Models: a truth value \( t, e, x, o, r, f \) to each atomic formula \( p \) at each state \( s_i \) of a linear discrete infinite sequence \( s_0, s_1, s_2, \cdots \)

Evaluate CPL formula: at state \( s_i \) using truth tables

Evaluate “temporal” formula: using relative order of states
Linear Temporal Logic (LTL)

Syntax: represents linguistic notions using logical connectives

**CPL**: captures “and”, “or”, “not”, “if then”, “iff”

**LTL**: adds “next”, “until”, “before”, “always”, “eventually”

\[
\text{atom} ::= p \mid q \mid r \mid \cdots
\]

\[
\varphi, \psi ::= \text{atom} \mid \neg \varphi \mid \varphi \land \psi \mid \varphi \lor \psi \mid \varphi \rightarrow \psi \mid \varphi \leftrightarrow \psi
\]

\[
\mid X \varphi \mid \varphi U \psi \mid \varphi B \psi \mid G \varphi \mid F \varphi
\]

**Models**: a truth value \( t \) exor \( f \) to each atomic formula \( p \) at each state \( s_i \) of a linear discrete infinite sequence \( s_0, s_1, s_2, \cdots \)

**Evaluate CPL formula**: at state \( s_i \) using truth tables

**Evaluate “temporal” formula**: using relative order of states

**Satisfiable**: true at some state in some sequence

**Valid**: true in all states of every sequence

**Lemma**: \( \varphi \) is valid if and only if \( \neg \varphi \) is unsatisfiable
Reasoning About Business Processes Using LTL

Business Process: finite linear discrete sequence $s_0, s_1, s_2, \ldots, s_n$ of states from start state $s_0$ to end state $s_n$

Rule 1: A risk assessment (ra) must be carried out for each request to open a bank account (ro)

$$G(ro \rightarrow F ra)$$
Reasoning About Business Processes Using LTL

**Business Process:** finite linear discrete sequence
\[ s_0, s_1, s_2, \ldots, s_n \] of states from start state \( s_0 \) to end state \( s_n \)

**Rule 1:** A risk assessment (ra) must be carried out for each request to open a bank account (ro)

\[ G(\text{ro} \rightarrow \text{F ra}) \]

**Rule 2:** A request to open a bank account (ro) is granted (rog) only if the risk is low (ral)

\[ G(\text{rog} \rightarrow \text{ral}) \]
Reasoning About Business Processes Using LTL

Business Process: finite linear discrete sequence
\(s_0, s_1, s_2, \cdots, s_n\) of states from start state \(s_0\) to end state \(s_n\)

Rule 1: A risk assessment (ra) must be carried out for each request to open a bank account (ro)

\[G(ro \rightarrow F ra)\]

Rule 2: A request to open a bank account (ro) is granted (rog) only if the risk is low (ral)

\[G(rog \rightarrow ral)\]

Rule 3: A due diligence assessment (dd) must be carried out for each request to open a bank account (ro)

\[G(ro \rightarrow F dd)\]
Reasoning About Business Processes Using LTL

Business Process: finite linear discrete sequence
   \( s_0, s_1, s_2, \ldots, s_n \) of states from start state \( s_0 \) to end state \( s_n \)

Rule 1: A risk assessment (ra) must be carried out for each request to open a bank account (ro)

\[ \mathbf{G}(ro \rightarrow F ra) \]

Rule 2: A request to open a bank account (ro) is granted (rog) only if the risk is low (ral)

\[ \mathbf{G}(rog \rightarrow ral) \]

Rule 3: A due diligence assessment (dd) must be carried out for each request to open a bank account (ro)

\[ \mathbf{G}(ro \rightarrow F dd) \]

Rule 4: If a person fails due diligence (ddf) then he or she must be blacklisted (bl)

\[ \mathbf{G}(ddf \rightarrow F bl) \]
SNOMED CT (diagnosis support and data exchange)

- caused by induces a relation between medical terms
- is a induces subset relationship
- industry standard in many countries (Australia, UK, . . .)

Need: a logic that can represent subclassing and relations
Description Logic and Extensions

Subclassing

ViralPneumonia $\sqsubseteq$ InfectiousPneumonia

“viral pneumonia is an infectious pneumonia”

Relations

ViralPneumonia $\sqsubseteq \exists \text{causedBy}.\text{Virus}$

“viral pneumonia is caused by a virus”

Reasoning: Tool Support for . . .

- logical consistency of (medical) hypotheses
- structured exchange of (medical) data

Research Questions: How About . . .

- Probabilistic Reasoning? Non-monotonic Conditionals?
- Practical Optimisation / Fragments with Low Complexity?
Research: Logic and Computation

Logic: design a new logic L for reasoning about ...

Calculi: find a new or better calculus for logic L and prove that it is correct and complete

Automated Deduction: find new procedures for deciding satisfiability or validity in logic L

Interactive Proof: verify properties of existing calculi/systems

Complexity: analyse computational complexity of new procedure

Extensions: can the calculus be extended with specific theories like equality and arithmetic?

Prototypes: implement a prototype of the new method

Optimisations: find and implement new optimisations or faster data structures and prove them correct

Experiments: compare prototype with existing ones

$$$: commercialise your technology
Applications of Logics in Other Areas

Artificial Intelligence: logics for reasoning about “belief”, “knowledge”, “desires”, “intentions”

Diagnosis: fault diagnosis in electricity networks

Constraints: methods for solving large sets of constraints

Software Verification: verification of the L4 micro-kernel

Ontologies: logics for knowledge representation e.g. Galen

Web Service Composition: synthesis of new services from given set of basic ones

Discrete Event Systems: how to capture the continuous dynamics of an aeroplane for example

Software Engineering: program verification and synthesis

Haskell: functional programming language based on logic
People in L&C: Name and Place of PhD

**ANU Staff**
- Clem Baker-Finch UTasmania
- Jeremy Dawson Sheffield
- Rajeev Goré Cambridge
- Dirk Pattinson Munich
- John Slaney ANU
- Qing Wang Kiel

**NICTA Staff**
- Peter Baumgartner Koblenz
- Michael Norrish Cambridge

**Current PhD students:** Daniel Alarcon, Mohammad Abdul Aziz, Joshua Bax, Jan Kuester, Hendra Gunadi, Zhe Hou, Joseph Chang

**Past PhD Students:** working at Oracle Research Canberra, Google Zurich, Vienna University of Technology, INRIA Paris, DSTO Adelaide, Google California, finance industry Sydney, Microsoft Redmond

**Future PhD students:** always welcome!
Want to find out more ...

**COMP2600**: Formal Methods for Software Engineering
**COMP2620**: Logic
**MATH3343**: Foundations of Mathematics (Honours)
**COMP4630**: Overview of Logic

**Logic Summer School**: [http://lss.cecs.anu.edu.au/](http://lss.cecs.anu.edu.au/)


**Independent Study**: can be designed for you with appropriate permissions just ask
For the curious ...

**Digital Circuits:** No, $A \land \neg C$ is not normal since $KB \land A \land \neg C$ is unsatisfiable

**Banking Rules:** are flawed because we forgot to say that blacklisted people should not be given bank accounts

**Security Protocols:** the protocol is flawed, see Needham Schroeder protocol on wikipedia (the bug was found using logical methods)

**Contact:** rajeev.gore@anu.edu.au