Temporal & Modal Logic

E. Allen Emerson

Presenter:
Aly Farahat

Acronyms

- TL: Temporal Logic
- BTL: Branching-time Logic
- LTL: Linear-Time Logic
- CTL: Computation Tree Logic
- PTL: Propositional Linear Temporal Logic
- PLTLF: PLTL with only future timeline structures
- PLTLP: PLTL with only past timeline structures
- PLTLB: PLTL with future and past timeline structures
- BMCP: Branching-Time Logic Model Checking Problem
- LMCP: Linear-Time Logic Model Checking Problem

Contents

- Temporal Logic
  - Overview
  - Classification
  - PLTL
    - Syntax
    - Semantics
    - Identities
- Concurrency
- Model Checking
Introduction

- Modal Logic: describes different worlds over which P might be True or False.
- Temporal Logic: a Modal Logic whose worlds are discrete states

Reasoning about Programs

- Hoare Logic:
  - Terminating Programs
  - Prove Post-Condition given Pre-Condition
  - Transformational Semantics – No notion of time
- Temporal Logic:
  - Concurrent Nonterminating Programs
  - Prove Properties about Programs valid during execution

Q: Why do we need logic for programs?

Classification

- Propositional vs. First-Order
- Global vs. Compositional
- Branching vs. Linear Time
- Points vs. Intervals
- Discrete vs. Continuous
- Past vs. Future
Propositional Linear Temporal Logic

• Syntax
  – Propositions
  – Connectives
  – Formulae
  – Temporal Operators
• Semantics
  – Time Structure
  – Formulae Interpretation

Syntax “Definitions”

• Gp: Always p
• Fp: Eventually p
• pUq: p until q
• Xp: Next p
• F p: Infinitely often p
• G p : Almost everywhere p

Syntax

• The Rules for forming Formulae
  – Every Atomic Proposition P is a Formula
  – If p and q are formulae, so are:
    – If p and q are formulae, so are:
      
      \[ p \land q \]
      \[ \neg p \]
      \[ p \lor q \]
    – If p and q are formulae, so are:
      
      \[ Xp \]
      \[ pUq \]
      \[ Fp \]
      \[ Gp \]

Q: How the other connectives are syntactically defined?
Syntactical Equivalences

$$False = p \land \neg p$$
$$True = \neg False$$
$$p \rightarrow q = \neg p \lor q$$
$$p \leftrightarrow q = (p \rightarrow q) \land (q \rightarrow p)$$
$$Fp = True \lor Up$$
$$Gp = \neg F\neg p$$
$$pBq = \neg \neg (pUq)$$

Semantics

- Structure
  - $$S$$: A set of States
  - $$x$$: An infinite sequence of states
  - $$L: S \rightarrow \text{PowerSet}(AP)$$ Truth assignment for atomic propositions at each state. AP is the set of atomic propositions.
  - Notation: $$x^i satisifies P$$ starting from $$s$$ to the future.

Q: How can we link program computations and structures in PLTL?

Semantics (cont’d)

- $$x^i satisifies P$$ iff $$P$$ is atomic proposition
- $$x^i \models p \land q$$ iff $$x^i \models p$$ and $$x^i \models q$$
- $$x^i \models p$$ iff it is not the case that $$x^i \models \neg p$$, $$p$$ and $$q$$ are formulae
- $$x^i \models pUq$$ iff $$\exists j (x^j \models q \land \forall k, i \leq k < j (x^k \models p))$$
- $$x^i \models Xp$$ iff $$x^{i+1} \models p$$

Q: Verify the syntactical equivalences for $$Gp$$ and $$Fp$$ for PLTL…
Semantic Equivalences

\[
& \vdash G \neg p \equiv \neg F p \\
& \vdash F \neg p \equiv \neg G p \\
& \vdash X \neg p \equiv \neg X p \\
& \vdash F \neg p \equiv \neg G p \\
& \vdash G \neg p \equiv \neg F p \\
& \vdash (\neg p) \lor q \equiv \neg(p \land q)
\]

Semantic Implications

\[
& \vdash p \Rightarrow F p \\
& \vdash G p \Rightarrow p \\
& \vdash X p \Rightarrow F p \\
& \vdash G p \Rightarrow X p \\
& \vdash G p \Rightarrow F p \\
& \vdash G p \Rightarrow X G p \\
& \vdash p \lor q \Rightarrow F q \\
& \vdash G q \Rightarrow F q
\]

More Identities

\[
& \vdash F F p = F p \\
& \vdash F F p = F p \\
& \vdash F F p = F p \\
& \vdash G F p = G p \\
& \vdash G G p = G p \\
& \vdash G G p = G p \\
& \vdash G F p = G p \\
& \vdash G F p = G F p = G G p \\
& \vdash G X G p = G F p = G G p \\
& \vdash G F p = G F p = G G p \\
& \vdash G X G p = G X G p = G G p \\
& \vdash G X G p = G X G p = G G p
\]
Identities for X

\[ \vdash X(p \lor q) \equiv (Xp \lor Xq) \]

\[ \vdash X(p \land q) \equiv (Xp \land Xq) \]

\[ \vdash X(p \Rightarrow q) \equiv (Xp \Rightarrow Xq) \]

\[ \vdash X(p \equiv q) \equiv (Xp \equiv Xq) \]

\[ \vdash XFp \equiv FXp \]

\[ \vdash XGp \equiv GXp \]

\[ \vdash (X(p \Rightarrow q)) \equiv X(p \Rightarrow Xq) \]

Variants of PLTL

- Strong and Weak Until
- Inclusion of Past Tense Operators
- Reflexive and non-Reflexive operators

Global and Initial Equivalences

- \( P \) and \( Q \) are globally equivalent iff for all linear time structures and for all times, \( P \) has the same truth value as \( Q \).

- \( P \) and \( Q \) are initially equivalent iff for all linear time structures and for the initial state 0, \( P \) has the same truth value as \( Q \).
Power of PLTLF and PLTLB

- Theorem 3.1: With respect to Global Equivalence, PLTLB is strictly more expressive than PLTLF

- Theorem 3.2: With respect to Initial Equivalence, PLTLB has the same expressive power as PLTLF

Concurrency

- Concurrency is modeled by Nondeterminism and Fairness

- Abstract Model:
  - M: a time structure
  - $\Phi_{\text{Start}}$: an atomic proposition corresponding to the starting states
  - $\Phi$: a temporal logic fairness property

Fairness Properties

- Weak Fairness:
  $\bigwedge_{\Phi \in \Pi} \left( \neg \text{enabled} \implies \neg \text{executed} \right)$

- Strong Fairness:
  $\Phi = \bigwedge_{\Phi \in \Pi} \left( \neg \text{enabled} \implies \neg \text{executed} \right)$

- Impartiality:
  $\Phi = \bigwedge_{\Phi \in \Pi} \bigvee \text{executed}$

Q: Is there an order on these fairness properties?
Concrete Models

- Constraints on the abstract Model:
  - Structure on State Space: Message Passing vs. Shared Memory.
  - Domain Constraints: Define variables domains (Integer ranges, Enumerations)
  - Transition Constrains: Define specific allowable instructions- Read/Write Restrictions

Memory Models

- Shared Memory: Shared variables and location counters define the state space
- Distributed Shared Memory: Writes granted only to Process having affinity to the shared variable.
- Message Passing: zero message buffer size; i.e. CSP.

Concurrent Programs and PLTL

- For the concurrent program
  \[ P = (M, \Phi_{Start}, \Phi) \]
  and a temporal logic formula \(\rho\),
  \(\rho\) holds true for \(P\) iff
  \[ \forall x, M, x \models ((\Phi_{Start} \land \Phi) \rightarrow \rho) \]
- Intuitively, for all possible time structures satisfying the initial and fairness conditions, they should satisfy \(\rho\).
Model Checking

- A decision problem:
  - $M$ is a finite temporal structure
  - $P$ is a temporal logic formula
  - Is $M \models P$?

Formal Definition

- BMCP:
  - $M$: a finite BTL structure
  - $p$: a BTL formula
  - For every state $s$ in $M$ where $p$ is satisfied, Label $s$ with $p$
- LMCP:
  - $M$: a finite LTL structure
  - $p$: an LTL formula
  - For every state $s$ in $M$ starting a full path satisfying $p$, Label $s$ with $Ep$

Interpretation

- In BMCP, it is sufficient to label the states with the BTL formula truth value, without checking full paths from each state
- In LMCP, it is necessary to check, for each state, the set of all possible paths and label it with $Ep$ if a path exists
Theorems

- LMCP for PLTL is polynomial-time reducible to SAT-PLTL
- LMCP for PLTL is P-space complete
- LMCP for PLTL(F) is NP-complete
- BMCP for CTL is deterministic polynomial time

Modalities are what matters

- Given a model checking algorithm for LTL, there is a model checking algorithm for the corresponding BTL of the same order of complexity
- "corresponding": having the same basic modalities
  
  Q: Is this result contradicting the fact that BMCP is in P?

Conclusions

- TL formulae express properties of concurrent programs
- Concurrent programs are represented by structures in TL
- Model Checking is an application of TL to concurrent programming
Thank you!