Branching Temporal Logics

CS5090-1

02/05/2008

Steven Vormwald

Contents

- Overview
- Syntax of CTL*
- Semantics of CTL*
- Useful Closure Properties
- Multiprocess Branching Temporal Logics

Overview

- Branching Temporal Logic allows us to reason about multiple possible futures by representing time as a directed graph, with each node representing a state.
- Possible futures (or pasts, depending on the chosen interpretation) are represented by nodes that are reachable from the current node in the graph.

Overview Structures

- M=(S,R,L)
  - S: Set of states
  - R: Total binary relation, a subset of SxS, representing possible state transitions
  - L: Labeling that gives an interpretation of all atomic proposition symbols at state s
- M=(S,X,L)
  - X, the set of *fullpaths* in M, where a *fullpath* is an infinite sequence of states $s_0 s_1 \ldots$ such that there is a transition from state s_i to state s_{i+1} for every natural number i

Overview – Directed Graphs

- While it is possible to work with any arbitrary directed graph, it is often useful to place certain restrictions on it:
  - *Acyclic* if there are no directed cycles in the graph.
    - Note that since we require all nodes to have at least one successor, all acyclic structures must have an infinite number of states.
  - *Tree-like* if it is acyclic and every node has at most 1 immediate predecessor.
  - A *Tree* if it is tree-like and there exists a unique node (the *root*) which has no predecessors, and is a predecessor to every other node in the graph.

Overview – Directed Graphs

- Note that we can transform most graphs (including every graph discussed in the paper) into a logically equivalent tree by unwinding it:
  - Create a structure M'=(S',R',L') by selecting an initial state , adding (s_0,0) to S', and continuously performing a depth-first search, adding (s_x, n) to S' when s is encountered at depth n along finite path x.

Syntax of CTL*

- Operators of CTL* include all the operators of linear temporal logic, and 2 new operators:
  - The unary operator *E*, which corresponds to “for some possible future, it is the case that...”
  - The unary operator *A*, which corresponds to “for all possible futures, it is the case that...”

Syntax of CTL*

- Formulae
  - State formulae
    - Interpreted at the current state
    - Used to decide properties of the current state
  - Path formulae
    - Interpreted along some path(s) starting at the current state
    - Used to decide properties of future states

Syntax of CTL* - State Formulae

- State formulae are defined as follows:
  - (S1) All atomic propositions *P* are state formulae
  - (S2) If *p* and *q* are state formulae, then so are *p^q* and ¬*p*
  - (S3) If *p* is a path formula, then *Ep* and *Ap* are state formulae

Syntax of CTL* - Path Formulae

- Path formulae are defined as follows:
  - (P1) If *p* is a state formula, then *p* is also a path formula
  - (P2) If *p* and *q* are path formulae, then so are *p^q* and ¬*p*
  - (P3) If *p* and *q* are path formulae, then so are *Xp* and *pUq*

Semantics of CTL*

- A formula is interpreted in the context of a structure M.
  - We will use M=(S,R,L), as defined earlier to describe the semantics of CTL*
  - $x=(s_0 s_1 s_2 ...)$ denotes a *fullpath* on M
  - $x^i=(s_i s_{i+1} s_{i+2} ...)$ denotes the i^{th} suffix of fullpath x
  - $M,s_0 \models p$ denotes state formula p is true in M at state s_0
  - $M,x \models p$ denotes path formula p is true in M of fullpath x

Semantics of CTL*

- (S1) $M,s_0 \models p$ iff p is an atomic proposition interpreted as true in state $L(s_0)$
- (S2a) $M,s_0 \models p \wedge q$ iff $M,s_0 \models p$ and $M,s_0 \models q$
- (S2b) $M,s_0 \models \neg p$ iff not $M,s_0 \models p$
- (S3a) $M,s_0 \models Ep$ iff there exists a fullpath $x=(s_0 s_1 s_2 \ldots)$ such that $M,x \models p$
- (S3b) $M,s_0 \models Ap$ iff $M,x \models p$ for all fullpaths $x=(s_0 s_1 s_2 \ldots)$

Semantics of CTL*

- (P1) $M,x \models p$ iff $M,s_0 \models p$
- (P2a) $M,x \models p \wedge q$ iff $M,x \models p$ and $M,x \models q$
- (P2b) $M,x \models \neg p$ iff not $M,x \models p$
- (P3a) $M,x \models pUq$ iff there exists an x^i such that $M,x^i \models q$ and for all i, $j<i \Rightarrow M,x^j \models p$
- (P3b) $M,x \models Xp$ iff $M,x^1 \models p$

Semantics of CTL*

- A formula *p* is *valid* if
  - *M,s*|=*p* for every structure M and state *s*, when *p* is a state formula
  - *M,x*|=*p* for every structure M and fullpath *x*, when *p* is a path formula
- A formula *p* is *satisfiable* if
  - *M,s*|=*p* for some structure M and state *s*, when *p* is a state formula
  - *M,x*|=*p* for some structure M and fullpath *x*, when *p* is a path formula

Useful Closure Properties

- It is often useful to work with structures where the set of *fullpaths* has some closure properties
  - Suffix Closure
  - Fusion Closure
  - Limit Closure
  - R-generable

Suffix Closure

- The set of fullpaths X on a structure is *suffix closed* if for every fullpath $x=(s_0 s_1 s_2 ...)$ in X, the fullpath $x^1=(s_1 s_2 s_3 ...)$ is also in X.
- Necessary to show some propositions, such as *EFp^¬p=>EXEFp*, are valid.

Fusion Closure

- The set of fullpaths X on a structure is *fusion closed* if for all fullpaths $x_0=(...s_u s s_v...)$ and $x_1=(...s_x s s_y...)$ in X, the fullpath $x_2=(...s_u s s_y...)$ (the prefix of x_0 up to s_u, followed by s, followed by the suffix of x_1 starting at s_y) is also in X.
- Necessary to show some propositions such as $EFEFp \equiv EFp$, are valid.

Limit Closure

- The set of fullpaths X on a structure is *limit closed* if whenever there exists a fullpath *x* such that all finite prefixes of that fullpath are prefixes of paths in X, *x* is also in X.
- Necessary to show some propositions, such as *p^AGEXp=>EGp*, are valid.

R-Generable

- The set of fullpaths X on a structure is *R-generable* if it is suffix, fusion, and limit closed.
- Most basic types of structures, notably those corresponding to program execution with nondeterministic scheduling, are *R-generable*.

Multiprocess Logics

- Extend the idea of a structure to M=(S,**R**,L), where **R** is a family of binary relations such that the union over **R** is a total binary relation that is a subset of SxS.
- Extend the idea of a fullpath such that between each state in the path, there is a label that notes which process (which member of **R**) executed the transition.

Multiprocess Logics

- Include propositions $enabled_j$ (atomic proposition) and $executed_j$ (atomic arc assertion) to represent the following formulas:
  - $M,s_o \models enabled_j$ iff s_o in domain R_j
    - Intuitively, it is possible for process *j* to cause a transition from the current state.
  - $M,x \models executed_j$ iff $d_1=j$
    - Intuitively, process *j* caused the transition from the current state in fullpath *x*.

Equivalence to Other Formalisms

- Transition systems

- Do-od systems

Generalized Multiprocess Logic

- Extend M again, such that M=(S,R,X,L) where
  - S is the set of states
  - R is a total binary relation, a subset of SxS
  - X is the set of fullpaths
  - L is a labeling function that associates states with interpretations of all atomic propositions, and each member of R interpretations of all arc assertions

Questions?

Question 1

- Can “processes” have private state information in the presented formalisms?

Question 2

- Can we reason about whether or not an arbitrary state can/must be entered from a given current state using the presented formalisms?

Question 3

- Can we reason about systems where different “observers” of the system see different orderings (paths) of transitions in the system using the presented formalisms?

Question 4

- Are the presented formalisms sufficient to describe concurrent program behavior?

Question 5

- How do these formalisms relate to formalisms such as Hoare's CSP that represent concurrency as independent communicating processes?