Temporal and Modal Logic

Based on paper:
E.A. Emerson.
Temporal and Modal Logic
J. van Leeuwen, editor, Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics, pages 995–1072, Elsevier, 1990.

Overview

1. Temporal and Modal Logic
2. Time
3. Propositional Linear Time Logic
4. Branching Time Logic (CTL and CTL*)
5. Model Checking
6. Concurrency
7. Kripke Structures and Verification of Programs

Temporal and Modal Logic

Modal logic originally developed by philosophers to study different "modes of truth", i.e. an assertion may be true depending on the given world.

Temporal logic (TL) is a special kind of modal logic where truth values of assertions vary with *time*.

Typical modalities (operators) are:

- "*sometimes* P": is true if P holds at some future moment
- "*always* P" is true if P holds at all future moments

Temporal logic is often used to specify and verify reactive systems, i.e. systems which continuously interact with the environment.

Time

TL-s can be classified by their view on 'time'

- Discrete versus continuous time

- Points versus intervals

- Linear time versus branching time

- Past versus future

- Finite versus infinite into the future

Linear Time Structures

Linear time is a totally ordered set $(S, <)$; discrete linear time is a countable totally ordered set, therefore isomorphic to $(\mathbb{N}, <)$.

For the moment, we consider discrete linear time.

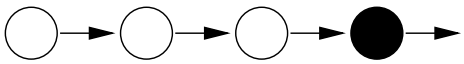
A linear time structure is a three tuple $M = (S, x, L)$, depending on a set AP of atomic propositions $\{P, Q, P', Q', \ldots\}$, of:

- a set S of *states*
- a *timeline* $x : \mathbb{N} \to S$
- a *labeling* $L : S \to \wp(AP)$ of states

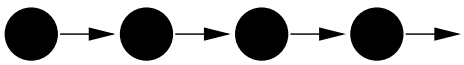
notation A timeline x is denoted as $s_0 s_1 \ldots$. Let $x = s_0 s_1 s_2 \ldots$. We write $x(j)$ for s_j, and x^j for $s_j s_{j+1} \cdots$

Propositional Linear Time Logic

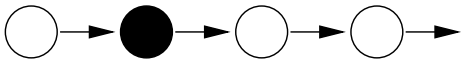
1. $\mathbf{F}p$ – sometimes p



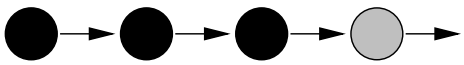
2. $\mathbf{G}p$ – always p



3. $\mathbf{X}p$ – nexttime p



4. $p\mathbf{U}q$ – p until q



PLTL syntax

definition

PLTL is least set of formulae generated by

1. each atomic proposition P is a formula

2. if p and q are formulae then $p \wedge q$ and $\neg p$ are formulae

3. if p and q are formulae then $p\mathbf{U}q$ and $\mathbf{X}p$ are formulae

Other temporal modalities are defined as abbreviations:

$\mathbf{F}p \equiv true\mathbf{U}p$ and $\mathbf{G}p \equiv \neg\mathbf{F}\neg p$

Also, the infinitary modalities:

$\mathbf{F}^{\infty}p \equiv \mathbf{GF}p$ and $\mathbf{G}^{\infty}p \equiv \mathbf{FG}p$

PLTL semantics

PLTL semantics defined with respect to a time structure and a time line

notation: Let $M = (S, x, L)$

$M, x \models p$, "in structure M and time line x formula p is true"

definition:

1. $x \models P$ iff $P \in L(x(0))$, for $P \in AP$

2. $x \models p \wedge q$ iff $x \models p$ and $x \models q$
   $x \models \neg p$ if not $x \models p$

3. $x \models (p \mathbf{U} q)$, iff $\exists_j x^j \models q$ and $\forall_{k<j} x^k \models p$
   $x \models \mathbf{X} p$ iff $x^1 \models p$

Satisfiability and Validity

A PLTL formula p is *satisfiable* iff there exists a linear time structure $M = (S, x, L)$ such that $M, x \models p$

A PLTL formula p is *valid*, notation $\models p$, iff for all linear time structures $M = (S, x, L)$ we have $M, x \models p$.

examples:

PLTL formula	satisfiable	valid
$p \Rightarrow \mathbf{F}q$	$\surd$	
$\mathbf{G}(p \Rightarrow \mathbf{X}q)$	$\surd$	
$p \wedge \mathbf{G}(p \Rightarrow \mathbf{X}p) \Rightarrow \mathbf{G}p$	$\surd$	$\surd$

Significant validities (i)

dualities

$$\models \mathbf{G}\neg p \equiv \neg\mathbf{F}p \qquad \models \mathbf{F}\neg p \equiv \neg\mathbf{G}p$$
$$\models \mathbf{X}\neg p \equiv \neg\mathbf{X}p$$
$$\models \mathbf{F}^{\infty}\neg p \equiv \neg\mathbf{G}^{\infty}p \qquad \models \mathbf{G}^{\infty}\neg p \equiv \neg\mathbf{F}^{\infty}p$$

implications

$$\models p \Rightarrow \mathbf{F}p \qquad \models \mathbf{G}p \Rightarrow p$$
$$\models \mathbf{X}p \Rightarrow \mathbf{F}p \qquad \models \mathbf{G}p \Rightarrow \mathbf{X}p$$
$$\models \mathbf{G}p \Rightarrow \mathbf{F}p \qquad \models \mathbf{G}p \Rightarrow \mathbf{X}\mathbf{G}p$$
$$\models p\mathbf{U}q \Rightarrow \mathbf{F}q \qquad \models \mathbf{G}^{\infty}q \Rightarrow \mathbf{F}^{\infty}q$$

Significant validities (ii)

idempotence

$$\models \mathbf{F}\mathbf{F}p \equiv \mathbf{F}p \qquad \models \mathbf{F}^{\infty}\mathbf{F}^{\infty}p \equiv \mathbf{F}^{\infty}p$$
$$\models \mathbf{G}\mathbf{G}p \equiv \mathbf{G}p \qquad \models \mathbf{G}^{\infty}\mathbf{G}^{\infty}p \equiv \mathbf{G}^{\infty}p$$

infinitary modalities

$$\begin{aligned} \models \quad & \mathbf{F}^{\infty}p \equiv \mathbf{X}\mathbf{F}^{\infty}p \equiv \mathbf{F}\mathbf{F}^{\infty}p \equiv \\ & \mathbf{G}\mathbf{F}^{\infty}p \equiv \mathbf{F}^{\infty}\mathbf{F}^{\infty}p \equiv \mathbf{G}^{\infty}\mathbf{F}^{\infty}p \\ \models \quad & \mathbf{G}^{\infty}p \equiv \mathbf{X}\mathbf{G}^{\infty}p \equiv \mathbf{F}\mathbf{G}^{\infty}p \equiv \\ & \mathbf{G}\mathbf{G}^{\infty}p \equiv \mathbf{F}^{\infty}\mathbf{G}^{\infty}p \equiv \mathbf{G}^{\infty}\mathbf{G}^{\infty}p \end{aligned}$$

Significant validities (iii)

distribution over boolean connectives

$$\begin{array}{l}
\models \mathbf{F}(p \vee q) \equiv (\mathbf{F}p \vee \mathbf{F}q) \\
\models \mathbf{F}^{\infty}(p \vee q) \equiv (\mathbf{F}^{\infty}p \vee \mathbf{F}^{\infty}q) \\
\models \mathbf{G}(p \wedge q) \equiv (\mathbf{G}p \wedge \mathbf{G}q) \\
\models \mathbf{G}^{\infty}(p \wedge q) \equiv (\mathbf{G}^{\infty}p \wedge \mathbf{G}^{\infty}q) \\
\models ((p \wedge q)\mathbf{U}r) \equiv ((p\mathbf{U}r) \wedge (q\mathbf{U}r)) \\
\models (p\mathbf{U}(q \vee r)) \equiv ((p\mathbf{U}q) \vee (p\mathbf{U}r))
\end{array}$$

$$\begin{array}{ll}
\models \mathbf{X}(p \vee q) \equiv \mathbf{X}p \vee \mathbf{X}q & \models \mathbf{X}(p \wedge q) \equiv \mathbf{X}p \wedge \mathbf{X}q \\
\models \mathbf{X}(p \Rightarrow q) \equiv \mathbf{X}p \Rightarrow \mathbf{X}q & \models \mathbf{X}(p \Leftrightarrow q) \equiv \mathbf{X}p \Leftrightarrow \mathbf{X}q
\end{array}$$

fixed point characterizations

$$\begin{array}{ll}
\models \mathbf{F}p \equiv p \vee \mathbf{XF}p & \models \mathbf{G}p \equiv p \wedge \mathbf{XG}p \\
\models p\mathbf{U}q \equiv q \vee (p \wedge \mathbf{X}(p\mathbf{U}q)) &
\end{array}$$

Other Variants of Linear Temporal Logic

Other variants of linear temporal logic can be constructed from PLTL by

1. also allowing finite time structures

2. changing the semantics of the modalities: for instance, change $\mathbf{U}$ into "$p\mathbf{U}q$ iff p holds as long as $\neg q$ holds" (weak until); or, change $\mathbf{U}$ into "$p\mathbf{U}q$ iff in a future moment (not now) q holds and until then p holds" (look at the strict future)

3. adding first-order or higher-order logic constructs (FOLTL)

4. adding past-tense temporal operators (PLTLP)

5. adding real-time

6. etc, etc, ...

Branching Temporal Logics

Time structures have a branching tree-like structure.

A *Kripke structure* is a triple $M = (S, R, L)$ where

- S is a set of *states*
- $R \subseteq S \times S$ is a total relation
- $L \in S \to \wp(AP)$ is a *labeling* of states

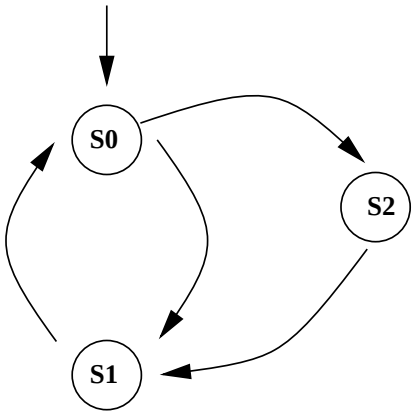
We say that M is

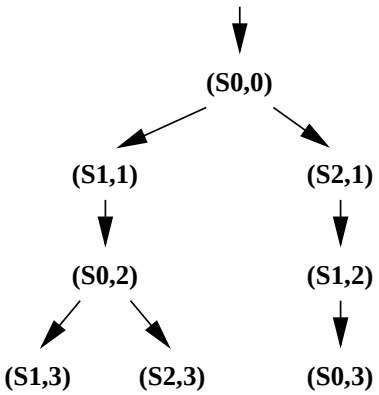
- *acyclic* iff there exists no directed cycles
- *tree-like* iff acyclic and each node has, at most one R-predecessor
- *a tree* iff tree-like and all nodes are reachable from a unique (root) node

Unwinding of graphs

A graph M starting from a state s_0 can be unwinded into a tree

example





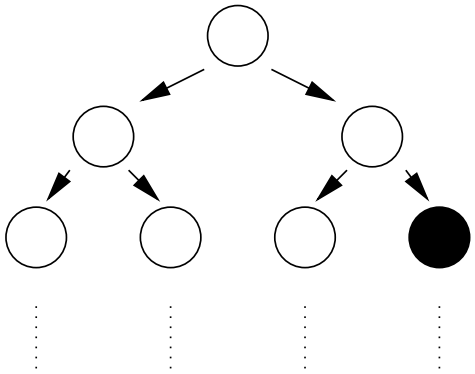
Propositional Branching Temporal Logics

We add *path* quantifiers

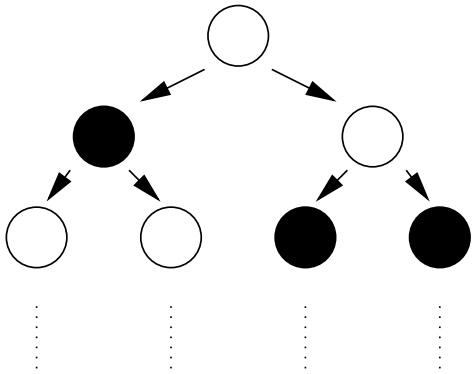
- **A**, where $\mathbf{A}p$ denotes that p holds over all paths
- **E**, where $\mathbf{E}p$ denotes that there exists some path such that p holds

And discuss the logics CTL (Computation Tree Logic) and the more expressive variant CTL*.

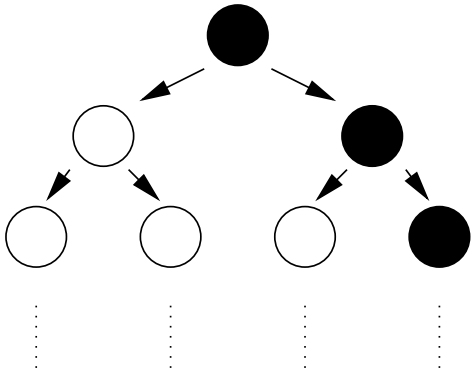
Basic CTL operators



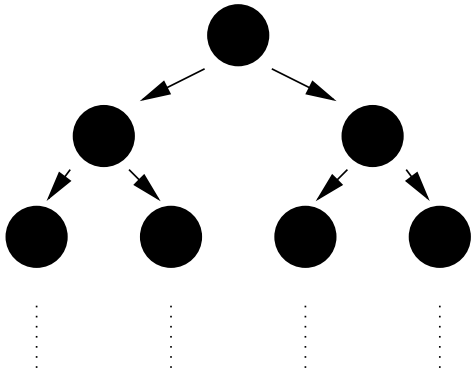
$\mathbf{EF}p$



$\mathbf{AF}p$



$\mathbf{EG}p$



$\mathbf{AG}p$

CTL* syntax

CTL* is least set of formulae generated by

(S1) each atomic proposition P is a state formula

(S2) if p and q are state formulae then so are $p \wedge q$, $\neg p$

(S3) if p is a path formula then $\mathbf{A}p$, $\mathbf{E}p$ are state formulae

(P1) all state formulae are path formulae

(P2) if p and q are path formulae then so are $p \wedge q$, $\neg p$

(P3) if p and q are path formulae then so are $p\mathbf{U}q$, $\mathbf{X}p$

The restricted language CTL replaces (P1-3) by

(P0) if p and q are state formulae then $p\mathbf{U}q$, $\mathbf{X}p$ are path formulae

CTL* semantics (i)

Semantics defined with respect to a structure M and a state s_0 or a *path* x

A *path* is an infinite sequence $s_0 s_1 \ldots$ where $\forall_i R(s_i, s_{i+1})$

Let x be the path $s_0 s_1 \ldots$, we write x^j for $s_j s_{j+1} \ldots$, and $x(j)$ for s_j

notation

- $M, s_0 \models p$: state formula p is true in M at s_0
- $M, x \models p$: path formula p is true in M of x

CTL* semantics (ii)

$\models$ is inductively defined as follows

(S1) $M, s_0 \models P$ iff $P \in L(s_0)$

(S2) $M, s_0 \models p \wedge q$ iff $M, s_0 \models p$ and $M, s_0 \models q$
$M, s_0 \models \neg p$ iff not $(M, s_0 \models p)$

(S3) $M, s_0 \models \mathbf{E}p$ iff $\exists path\ x : x(0) = s_0 \wedge M, x \models p$
$M, s_0 \models \mathbf{A}p$ iff $\forall path\ x : x(0) = s_0 \Rightarrow M, x \models p$

(P1) $M, x \models p$ iff $M, x(0) \models p$

(P2) $M, x \models p \wedge q$ iff $M, x \models p$ and $M, x \models q$
$M, x \models \neg p$ iff not $(M, x \models p)$

(P3) $M, x \models p\mathbf{U}q$ iff $\exists_i M, x^i \models q$ and $\forall_{j<i} M, x^j \models p$
$M, x \models \mathbf{X}p$ iff $M, x^1 \models p$

Model Checking

Given a finite structure M and a TL formula p:

Does M model p???

Lemma 20.1. *Model checking for PLTL is PSPACE-complete*

Lemma 20.2. *Model checking for CTL is in deterministic polynomial time*

Lemma 20.3. *Model checking for CTL* is PSPACE-complete*

Concurrency (i)

observe

two processes P_1 and P_2, what if they are ran in parallel?

We expect

$$P_1P_2P_1P_2, \ldots \text{ or } P_1P_1P_2P_1P_1P_2 \text{ or even } P_1P_1P_2P_2P_1P_1 \ldots$$

but not

$$P_1P_1P_1P_1 \ldots \text{ or } P_2P_2P_2 \ldots \text{ or even } P_1P_2P_2P_2P_2 \ldots$$

How to model concurrency?

Concurrency (ii)

Multi-process structures are the products of several structures

Fairness is modeled by fair scheduling assumptions described as TL formula over the processes

definition

According to TL:

$$\text{concurrency} = \text{nondeterminism} + \text{fairness}$$

Concurrency (iii)

Typical fairness assumptions: assume $P_1 \ldots P_k$

1. *unconditional fairness*

$$\wedge_{i=1}^{k} \mathbf{F}^{\infty} executed$$

2. *weak fairness*

$$\wedge_{i=1}^{k} \mathbf{G}^{\infty} enabled \Rightarrow \mathbf{F}^{\infty} executed$$

3. *strong fairness*

$$\wedge_{i=1}^{k} \mathbf{F}^{\infty} enabled \Rightarrow \mathbf{F}^{\infty} executed$$

Fair Kripke Structures

Büchi: an infinite path $\pi = s_0 s_1 \ldots$ is *fair* with respect to a partitioning F on states iff forall $P \in F$ there exists a state $s \in P$ which occurs infinitely often in π.

A *Kripke structure* with *fairness constraints* is a tuple (S, S_0, M, AP, L, R, F) where

- S is a finite set of *states*
- $S_0 \subseteq S$ is a set of *start states*
- $M \subseteq S \times S$ is set of *edges*
- AP is a set of *atomic propositions*
- $L \in S \to \wp(AP)$ is a *labeling* of states
- $F \subseteq \wp(S)$ is a set of states with Büchi fairness assumptions.

Verification of Concurrent Programs

```
MODULE main
VAR
  gate1 : process inverter(gate3.output);
  gate2 : process inverter(gate1.output);
  gate3 : process inverter(gate2.output);
SPEC
  (AG AF gate1.output) & (AG AF !gate1.output)

MODULE inverter(input)
VAR
  output : boolean;
ASSIGN
  init(output) := 0;
  next(output) := !input;
FAIRNESS
  running
```

And the output

```
-- specification AG AF gate1.output
                 & AG AF (!gate1.outpu...
   is true

resources used:
user time: 0.4 s, system time: 0.55 s
BDD nodes allocated: 239
Bytes allocated: 917504
BDD nodes representing
    transition relation: 32 + 1
```