

Logic Summer School, ANU Dec 2003
Formal Methods 1: Algorithmic Verification
Lectures 2 & 3: Linear Time Temporal Logic, Buchi
automata
Ron van der Meyden (UNSW/NICTA)

- This lecture:
1. Introduction to (Future) Linear Time temporal logic, one of the commonly used temporal logics in computer science.*
 2. Introduction to *one way* to approach verification of concurrent programs using temporal logic: Buchi automata
 3. LTL/Buchi automata in SPIN

Propositional Temporal Logic - Syntax

Let *Prop* be a set of *atomic propositions*. The formulae of temporal logic are defined as follows:

1. If $P \in Prop$ then P is a formula.
2. If φ is a formula then $\neg\varphi$ is a formula.
3. If φ_1 and φ_2 are formulae then $\varphi_1 \wedge \varphi_2$ is a formula.
4. If φ is a formula then $X\varphi$ is a formula, intuitively stating that φ is true at the next moment of time.
5. If φ_1 and φ_2 are formulae then $\varphi_1 \mathbf{U} \varphi_2$ is a formula, intuitively stating that φ_2 is eventually true, and φ_1 will be true until φ_2 is true.

Some defined operators:

1. $\mathbf{F}\varphi$ is defined to be **true** $\mathbf{U} \varphi$. This states that φ is eventually true.
2. $\mathbf{G}\varphi$ is defined to be $\neg\mathbf{F}\neg\varphi$. This states that φ is true now and at all future times.

Example: Mutual Exclusion

ϕ is the conjunction of the following formulas:

1. (Mutual Exclusion) $\neg \mathbf{F}(loc_1 = \text{crit} \wedge loc_2 = \text{crit})$
2. (Liveness) $\bigwedge_{i=1,2} \mathbf{G}(loc_i = \text{try} \Rightarrow \mathbf{F}(loc_i = \text{crit}))$
3. For each transition t of process i , let $enabled(t)$ be a formula that captures the circumstances in which t is enabled.

Example: for the transition t from `try` to `crit` of process 0, we have $enabled(t) = loc_0 = \text{try} \wedge (flag[1] == 0 \vee last == 1)$

(Deadlock freedom) $\mathbf{G} \bigvee_{t \in \text{trans}(P_0) \cup \text{trans}(P_1)} enabled(t)$

Propositional Temporal Logic - Semantics

The temporal logic is interpreted over structures called runs. We assume an underlying set *State* of states has been defined. Typically these states will arise from the possible states of a computation.

A *run* or *model* is an infinite sequence r of states:

$$r = s_0, s_1, s_2, s_3, \dots$$

We write $r(n)$ for the state s_n .

We assume a relation \models of satisfaction of basic propositions in a state has been defined. We write $s \models P$ if proposition P is true in state s .

Satisfaction of a formula φ of temporal logic at a time n in a run r is defined inductively as follows:

$(r, n) \models P$	if $r(n) \models P$, for $P \in Prop$
$(r, n) \models \neg \varphi$	if not $(r, n) \models \varphi$.
$(r, n) \models \varphi_1 \wedge \varphi_2$	if $(r, n) \models \varphi_1$ and $(r, n) \models \varphi_2$.
$(r, n) \models \mathbf{X}\varphi$	if $(r, n+1) \models \varphi$.
$(r, n) \models \varphi_1 U \varphi_2$	if there exists $m \geq n$ with $(r, m) \models \varphi_2$ and $(r, k) \models \varphi_1$ for all k with $n \leq k < m$.

Validity

A formula φ is said to be *valid* (with respect to the *floating* semantics) if $(r, n) \models \varphi$ for all runs r and all times n .

A formula φ is said to be *valid* (with respect to the *anchored* semantics) if $(r, 0) \models \varphi$ for all runs r .

We present a logic that characterizes both notions of validity.

As usual, a formula φ is a *theorem* of the temporal logic, written $\vdash \varphi$, if there exists a sequence

$$\varphi_0, \varphi_1, \dots, \varphi_m$$

such that $\varphi = \varphi_m$ and each φ_i is either an axiom or follows from $\varphi_0, \dots, \varphi_{i-1}$ using a rule of inference.

Axioms

T0. All tautologies of propositional logic.

T1. $\mathbf{X}(\varphi) \wedge \mathbf{X}(\varphi \Rightarrow \psi) \Rightarrow \mathbf{X}\psi$

T2. $\mathbf{X}(\neg\varphi) \Leftrightarrow \neg\mathbf{X}\varphi$

T3. $\varphi \mathbf{U} \psi \Leftrightarrow \psi \vee (\varphi \wedge \mathbf{X}(\varphi \mathbf{U} \psi))$

Rules of Inference

MP. If φ_1 and $\varphi_1 \Rightarrow \varphi_2$ then φ_2 .

RT1. If φ then $\mathbf{X}\varphi$

RT2. If $\varphi' \Rightarrow \neg\psi \wedge \mathbf{X}\varphi'$ then $\varphi' \Rightarrow \neg(\varphi \mathbf{U} \psi)$

A run of a transition system $M = \langle S, S_0, \rightarrow, L \rangle$ is a sequence $r = s_0, s_1, \dots$ such that $s_0 \in S_0$ and $s_i \rightarrow s_{i+1}$ for all $i \geq 0$.

Given a concurrent program P and a formula ϕ of LTL, say that P *satisfies* ϕ , written $P \models \phi$, if for all runs r of the transition system M_P , we have $(r, 0) \models \phi$.

The verification problem: given P and ϕ , determine if $P \models \phi$

(Pnueli's methodology (as elaborated the books by Manna & Pnueli) for using temporal logic to prove $P \models \phi$:

1. Translate P into a formula Ψ_P of LTL such that $(r, 0) \models \Psi_P$ iff r is a run generated by P .
2. Prove that $\Psi_P \Rightarrow \psi$ is valid.

Fact: Determining validity of LTL formulas is PSPACE complete.

Finite State Automata

A non-deterministic finite state automaton is a tuple $A = \langle \Sigma, S, S_0, \rho, F \rangle$ where

1. Σ is an alphabet
2. S is a set of states
3. S_0 a subset of S (the *initial* states)
4. $\rho : S \times \Sigma \rightarrow \mathcal{P}(S)$ a transition function
5. F a subset of S (the *final* states)

An *input* to A is a finite sequence $w = a_0, a_1, \dots, a_{n-1}$, where each $a_i \in \Sigma$.

A *run* of A on input w is a sequence $r = s_0, s_1, \dots, s_n$ such that

1. $s_0 \in S_0$
2. $s_{i+1} \in \rho(s_i, a_i)$ for $i = 0 \dots n - 1$

A run r is *accepting* if $s_n \in F$.

The *language* accepted by A is the set $L(A)$ of words $w \in \Sigma^*$ such that there exists an accepting run of A on input w .

Büchi automata

A Büchi automaton is a finite state automaton $A = \langle \Sigma, S, S_0, \rho, F \rangle$ used to accept languages consisting of *infinite strings*:

An *input* to A is an infinite sequence $w = a_0, a_1, \dots \in \Sigma^\omega$.

A *run* of A on input w is an infinite sequence $r = s_0, s_1 \dots \in S^\omega$ such that

1. $s_0 \in S_0$
2. $s_{i+1} \in \rho(s_i, a_i)$ for all i

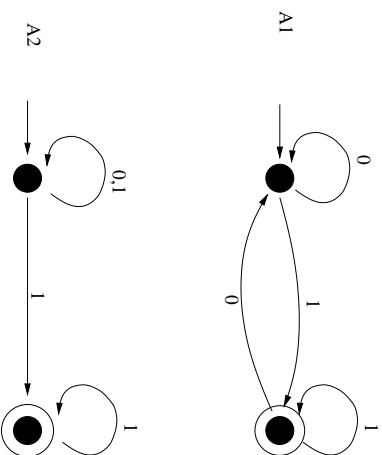
Acceptance for Buchi Automata

Given a run $r = s_0, s_1, \dots$, define $\text{inf}(r)$ to be the set of states that occur *infinitely often* in r .

A run r is *accepting* if $\text{inf}(r) \cap F$ is nonempty.

The *language* accepted by A is the set $L(A)$ of words $w \in \Sigma^\omega$ such that there exists an accepting run of A on input w .

Examples:



$$L(A1) = (\Sigma^*1)^\omega$$

$$L(A2) = \Sigma^*1^\omega$$

Emptiness for Buchi automata:

Problem: given a Buchi automaton A , determine if $L(A)$ is empty.

Given A , construct the graph $G(A) = (S, E)$ where S is the set of states of A and $(s, t) \in E$ iff there exists a letter $a \in \Sigma$ such that $t = \rho(s, a)$.

Theorem: $L(A)$ is empty iff there exists a finite path

1. $s_0 \in S_0$, and
2. $s_n = s_k$ (and $n > k$), and
3. $\{s_k, \dots, s_n\} \cap F \neq \emptyset$.

Given a graph $G = (V, E)$ and a set $P \subseteq V$:

1. P is strongly connected if for every pair $s, t \in P$, there exists a path from s to t
2. P is a *maximally strongly connected component* (MSCC) of G if P is strongly connected and no larger subset of V is strongly connected.

(Note: a single node with no self-edge is not considered to be strongly connected.)

Fact (Tarjan): using depth first search, a graph can be decomposed into MSCC's in linear time

Generalized Büchi automata

A *generalized Büchi automaton* is a Büchi automaton, with instead of a set $F \subseteq S$ of acceptance states, a *collection* $\{F_1, \dots, F_k\}$ of sets

$$F_k \subseteq S.$$

A run r is accepted if $\inf(r) \cap F_i \neq \emptyset$ for all $i = 1 \dots k$.

Exercise: Show that the change to multiple acceptance sets does not increase expressive power, by showing that for each generalized Büchi automaton A , we can construct a Büchi automaton A' such that $L(A) = L(A')$.

An algorithm for testing emptiness of a Büchi automaton

1. Decompose the graph $G(A)$ into MSCC's, starting from S_0
2. Check that at least one MSCC has non-empty intersection with F , if so return("non-empty"), else return("empty")

This can be done in time linear in the number of states and edges.

In practice, a simpler algorithm is used:

Double Depth First Search

```
DFS1(q)
  hash(q);
  for all successors q' of q
    do if q' not in hash table then DFS1(q');
    if accept(q) then DFS2(q)
DFS2(q)
  flag(q);
  for all successors q' of q
    do if q' on DFS1 stack then TERMINATE(true)
    else if q' not flagged then DFS2(q')
```

Applying Büchi automata to test validity of LTL formulas

Theorem: for every LTL formula ϕ over basic propositions $Prop$, we can construct a Büchi automaton A_ϕ with alphabet $\mathcal{P}(Prop)$ such that $(r; 0) \models \phi$ iff $r \in L(A_\phi)$.

(Here states in r are subsets of $Prop$, and for $p \in Prop$, we take $s \models p$ if $p \in s$.)

So ϕ is valid iff $\neg\phi$ is not satisfiable iff there are no runs r such that $(r; 0) \models \neg\phi$ iff $L(A_{\neg\phi})$ is empty.

Intersection of Büchi automata

Propn: Let $A = \langle \Sigma, S, S_0, \rho, F \rangle$ $A' = \langle \Sigma, S', S'_0, \rho', F' \rangle$ be Büchi automata. Then $L(A) \cap L(A') = L(B)$, where $B = \langle \Sigma, S^*, S_0^*, \rho^*, F^* \rangle$ is the generalized Büchi automaton with

1. $S^* = S \times S'$
2. $S_0^* = S_0 \times S'_0$
3. $\rho^*((s, s'), a) = \rho(s, a) \times \rho'(s', a)$
4. $F^* = \{F \times S', S \times F'\}$

There are similar operations on Büchi automata for \neg , **X**, **U**.

However, the construction for \neg involves an exponential blowup in the number of states (subset construction).

A tableaux approach turns out to be more efficient for LTL formulas....

Given a formula ϕ of LTL, define the *closure* $cl(\phi)$ to be the set of all subformulas of ϕ and their negations (treat $\neg\neg\alpha$ as α).

Example: If $\phi = p \mathbf{U} (\mathbf{X}(\neg q))$ then

$$cl(\phi) = \{p \mathbf{U} (\mathbf{X}(\neg q)), \neg(p \mathbf{U} (\mathbf{X}(\neg q))), p, \neg p, \mathbf{X}(\neg q), \neg\mathbf{X}(\neg q), \neg q, q\}$$

Say that $s \subseteq cl(\phi)$ is an *atom* for ϕ if

1. for each $\alpha \in cl(\phi)$, either $\alpha \in s$ or $\neg\alpha \in s$
2. $\neg\phi_1 \in s$ iff $\phi_1 \notin s$
3. $\phi_1 \wedge \phi_2 \in s$ iff $\phi_1 \in s$ and $\phi_2 \in s$

Let w be a word over $\mathcal{P}(Prop)$.

A *Hintikka* sequence for w and ϕ is a sequence s_0, s_1, \dots where each s_i is an atom for ϕ such that $\phi \in s_0$ and for all $i \geq 0$,

1. $p \in s_i$ iff $p \in w(i)$, for all $p \in Prop \cap cl(\phi)$
2. $\mathbf{X}\phi_1 \in s_i$ iff $\phi_1 \in s_{i+1}$
3. $\phi_1 \mathbf{U} \phi_2 \in s_i$ iff $\phi_2 \in s_i$ or both $\phi_1 \in s_i$ and $\phi_1 \mathbf{U} \phi_2 \in s_{i+1}$
4. if $\phi_1 \mathbf{U} \phi_2 \in s_i$ then for some $j \geq i$ we have $\phi_2 \in s_j$

For a run w and time n , define

$$S(w, \phi, n) = \{\alpha \in cl(\phi) \mid (w, n) \models \alpha\}$$

$$S(w, \phi) = S(w, \phi, 0), S(w, \phi, 1), S(w, \phi, 2), \dots$$

Theorem: s_0, s_1, \dots is a Hintikka sequence for w and ϕ iff $s_0, s_1, \dots = S(w, \phi)$

Given ϕ , we construct a *generalized* Büchi automaton A_ϕ with

1. Alphabet $\Sigma = \mathcal{P}(Prop)$
2. States S : the set of atoms for ϕ
3. Initial states S_0 : states $s \in S$ such that $\phi \in s$.
4. Final state sets: $\{F_\alpha \cup \beta \mid \alpha \cup \beta \in cl(\phi)\}$ where

$$F_\alpha \cup \beta = \{s \in S \mid \alpha \cup \beta \notin s \text{ or } \beta \in s\}$$

and $\rho(s, a) = \emptyset$ if $a \cap Prop_\phi \neq s \cap Prop$, else $\rho(s, a)$ is the set of states s' such that

1. $\mathbf{X}\phi_1 \in s$ iff $\phi_1 \in s'$ (when $\mathbf{X}\phi_1 \in cl(\phi)$), and
2. $\phi_1 \cup \phi_2 \in s$ iff either $\phi_2 \in s$ or $\phi_1 \in s$ and $\phi_1 \cup \phi_2 \in s'$ (when $\phi_1 \cup \phi_2 \in cl(\phi)$)

Theorem: A_ϕ accepts a word w over $\mathcal{P}(Prop)$ iff $(w, 0) \models \phi$.

Corollary: ϕ is satisfiable iff $L(A_\phi)$ is empty.

Given a transition system $M = \langle S, S_0, \rightarrow, L \rangle$, we can construct a Büchi automaton A_M with alphabet $\mathcal{P}(Prop)$ such that $L(A_M)$ is equal to the set $\{L(s_0), L(s(1)), \dots \mid s_0, s_1, \dots \text{ is a run of } M\}$.

$$M \models \phi \text{ iff } L(A_M) \cap L(A_{\neg\phi}) = \emptyset$$

The latter condition can be tested by applying double DFS to the intersection automaton.