Linear Temporal Logic and Büchi Automata

(Based on [Manna and Pnueli 1992, 1995] and [Clarke et al. 1999])

Yih-Kuen Tsay

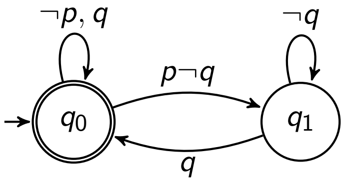
Department of Information Management
National Taiwan University

Outline

Introduction

- We have seen how automata, in particular Büchi automata, may be used to describe the behaviors of a concurrent system.
- Büchi automata "localize" temporal dependency between occurrences of events (represented by propositions) to relations between states and tend to be of lower level.
- We will study an alternative formalism, namely linear temporal logic.
- Temporal logic formulae describe temporal dependency without explicit references to time points and are in general more abstract.

Introduction (cont.)



- The above Büchi automaton says that, whenever p holds at some point in time, q must hold at the same time or will hold at a later time.
  Note: the alphabet is $\{pq, p\neg q, \neg pq, \neg p\neg q\}$; q alone represents any input symbol from $\{pq, \neg pq\}$.
- It may not be easy to see that this indeed is the case.
- In linear temporal logic, this can easily be expressed as $\Box(p \rightarrow \Diamond q)$, which reads "always p implies eventually q".

PTL: The Future

- We first look at the future fragment of Propositional Temporal Logic (PTL).
- Future operators include $\bigcirc$ (next), $\Diamond$ (eventually), $\Box$ (always), $\mathcal{U}$ (until), and $\mathcal{W}$ (wait-for).
- With $\mathcal{W}$ replaced by $\mathcal{R}$ (release), this fragment is often referred to as LTL (linear temporal logic) in the model checking community.
- Let V be a set of boolean variables.
- The future PTL formulae are defined inductively as follows:
  - Every variable $p \in V$ is a PTL formula.
  - If f and g are PTL formulae, then so are $\neg f$, $f \vee g$, $f \wedge g$, $\bigcirc f$, $\Diamond f$, $\Box f$, $f \; \mathcal{U} \; g$, and $f \; \mathcal{W} \; g$.
    ($\neg f \vee g$ is also written as $f \rightarrow g$ and $(f \rightarrow g) \wedge (g \rightarrow f)$ as $f \leftrightarrow g$.)
- Examples: $\Box(\neg C_0 \vee \neg C_1)$, $\Box(T_1 \rightarrow \Diamond C_1)$.

PTL: The Future (cont.)

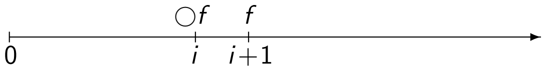
- A PTL formula is interpreted over an infinite sequence of states $\sigma = s_0 s_1 s_2 \cdots$, relative to a position in that sequence.
- A state is a subset of V, containing exactly those variables that evaluate to true in that state.
- If each possible subset of V is treated as a symbol, then a sequence of states can also be viewed as an infinite word over 2^V.
- The semantics of PTL in terms of $(\sigma, i) \models f$ (f holds at the i-th position of σ) is given below.
- We say that a sequence σ satisfies a PTL formula f or σ is a model of f, denoted $\sigma \models f$, if $(\sigma, 0) \models f$.

PTL: The Future (cont.)

- For a boolean variable p,
  - $(\sigma, i) \models p \iff p \in s_i$
- For boolean operators,
  - $(\sigma, i) \models \neg f \iff (\sigma, i) \models f$ does not hold
  - $(\sigma, i) \models f \vee g \iff (\sigma, i) \models f$ or $(\sigma, i) \models g$
  - $(\sigma, i) \models f \wedge g \iff (\sigma, i) \models f$ and $(\sigma, i) \models g$

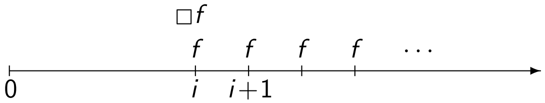
PTL: The Future (cont.)

- For future temporal operators,
  - $(\sigma, i) \models \bigcirc f \iff (\sigma, i+1) \models f$



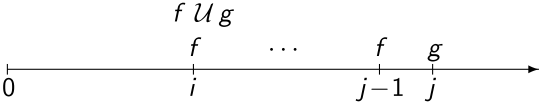
  - $(\sigma, i) \models \Diamond f \iff$ for some $j \geq i$, $(\sigma, j) \models f$

  - $(\sigma, i) \models \Box f \iff$ for all $j \geq i$, $(\sigma, j) \models f$



PTL: The Future (cont.)

- For future temporal operators (cont.),
  - $(\sigma, i) \models f\ \mathcal{U}\ g \iff$ for some $j \geq i$, $(\sigma, j) \models g$ and for all k, $i \leq k < j$, $(\sigma, k) \models f$

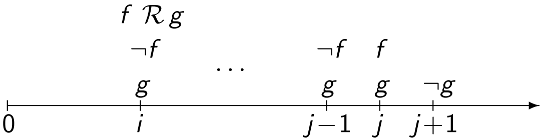


  - $(\sigma, i) \models f\ \mathcal{W}\ g \iff$ (for some $j \geq i$, $(\sigma, j) \models g$ and for all k, $i \leq k < j$, $(\sigma, k) \models f$) or (for all $k \geq i$, $(\sigma, k) \models f$)

    $f\ \mathcal{W}\ g$ holds at position i if and only if $f\ \mathcal{U}\ g$ or $\Box f$ holds at position i.

PTL: The Future (cont.)

- For future temporal operators (cont.),
  - When $\mathcal{R}$ is preferred over $\mathcal{W}$,
    $(\sigma, i) \models f \,\mathcal{R}\, g \iff$ for all $j \geq i$, if $(\sigma, k) \not\models f$ for all k, $i \leq k < j$, then $(\sigma, j) \models g$.



Simple On-the-Fly Translation

- We will study a tableau-based algorithm [GPVW] for obtaining a Büchi automaton from a PTL formula.
- The algorithm is geared towards being used in model checking in an on-the-fly fashion:
  It is possible to detect that a property does not hold by only constructing part of the model and of the automaton.
- The algorithm can also be used to check the validity of a temporal logic assertion.
- To apply the translation algorithm, we first convert the formula φ into the *negation normal form*.

Preprocessing of Formulae

Every LTL formula can be converted into the negation normal form:

- $\neg(p \wedge q) = (\neg p) \vee (\neg q)$
- $\neg(p \vee q) = (\neg p) \wedge (\neg q)$
- $\Diamond p$ (or $\mathbf{F}p$) $= \mathit{True}\ \mathcal{U}\ p$
- $\Box p$ (or $\mathbf{G}p$) $= \mathit{False}\ \mathcal{R}\ p$
- $\neg(p\ \mathcal{U}\ q) = (\neg p)\ \mathcal{R}\ (\neg q)$
- $\neg(p\ \mathcal{R}\ q) = (\neg p)\ \mathcal{U}\ (\neg q)$
- $\neg\bigcirc p$ (or $\neg\mathbf{X}p$) $= \bigcirc\neg p$

Note: "$p\ \mathcal{W}\ q$" was not treated in the original on-the-fly translation algorithm; $\neg(p\ \mathcal{W}\ q) \cong (\neg q)\ \mathcal{U}\ (\neg p \wedge \neg q)$.

Data Structure of an Automaton Node

- *ID*: a string that identifies the node.
- *Incoming*: the incoming edges, represented by the IDs of the nodes with an outgoing edge leading to this node.
- *New*: a set of subformulae that must hold at this state and have not yet been processed.
- *Old*: the subformulae that must hold at this state and have already been processed.
- *Next*: the subformulae that must hold in all states that are immediate successors of states satisfying the formulae in *Old*.

The Algorithm: Start and Overview

- Start with a single node having a single incoming edge labeled *init* (i.e., from an initial node).
- The starting node has initially one obligation in *New*, namely φ, and *Old* and *Next* are initially empty.
- Expand the starting node (which generates new nodes) in an *DFS* manner.
- Fully processed nodes are put in a list called *Nodes*.

$$
\begin{array}{l}
\textbf{function } \mathit{create\_graph}(\varphi) \\
\quad \mathit{expand}([\mathit{ID} \leftarrow \mathit{new\_ID}(), \\
\qquad\qquad \mathit{Incoming} \leftarrow \{\mathit{init}\}, \\
\qquad\qquad \mathit{Old} \leftarrow \emptyset, \\
\qquad\qquad \mathit{New} \leftarrow \{\varphi\}, \\
\qquad\qquad \mathit{Next} \leftarrow \emptyset], \\
\qquad\qquad\qquad\qquad \emptyset); \\
\textbf{end function}
\end{array}
$$

The Algorithm: Node-Expansion

- Check if there are unprocessed obligations in *New* of the current node N.
- If *New* is empty, it means node N is fully processed and ready to be added to *Nodes*.
- Otherwise, a formula in *New* is selected, processed, and moved to *Old*.

function $expand(q, Nodes)$
  if $New(q) = \emptyset$ **then**
    if $\exists r \in Nodes : Old(r) = Old(q) \wedge Next(r) = Next(q)$ **then**
      $\ldots$
    else $\ldots$
  else let $\eta \in New(q)$;
    $New(q) := New(q) - \eta$;
    $\ldots$
end function

The Algorithm: Node-Expansion (cont.)

/* in **function** *expand* */
if $New(q) = \emptyset$ **then**
    if $\exists r \in Nodes : Old(r) = Old(q) \wedge Next(r) = Next(q)$ **then**
        $Incoming(r) := Incoming(r) \cup Incoming(q)$;
        return$(Nodes)$;
    else $expand([ID \leftarrow new\_ID(),$
        $Incoming \leftarrow \{ID(q)\},$
        $Old \leftarrow \emptyset,$
        $New \leftarrow Next(q),$
        $Next \leftarrow \emptyset], \quad Nodes \cup \{q\})$;
    end if
else let $\eta \in New(q)$;
    $New(q) := New(q) - \eta$;
    if $\eta \in Old(q)$ **then** $expand(q, Nodes)$;
    else . . . /* cases according to the form of η */

The Algorithm: Updating the Nodes List

A fully processed current node N is added to *Nodes* as follows:

- If there already is a node in *Nodes* with the same obligations in both its *Old* and *Next* fields, the incoming edges of N are incorporated into those of the existing node.
- Otherwise, the current node N is added to *Nodes*.
- With the addition of node N in *Nodes*, a new current node is formed for its successor as follows:
  1. There is initially one edge from N to the new node.
  2. *New* is set initially to the *Next* field of N.
  3. *Old* and *Next* of the new node are initially empty.

The Algorithm: Node-Expansion (cont.)

A formula η in *New* is processed as follows:

- If η is just a literal (a proposition or the negation of a proposition), then
  - if $\neg\eta$ is in *Old*, the current node is discarded;
  - otherwise, η is added to *Old*.
- If η is not a literal, the current node can be split into two or not split, and new formulae can be added to the fields *New* and *Next*.
- The exact actions depend on the form of η.

The Algorithm: Node-Expansion (cont.)

$$
\begin{array}{l}
\textbf{case } \eta \textbf{ of} \\
\quad p \wedge q\text{:}\; q' := [\mathit{ID} \leftarrow \mathit{new\_ID}(), \\
\qquad\qquad\qquad \mathit{Incoming} \leftarrow \mathit{Incoming}(q), \\
\qquad\qquad\qquad \mathit{Old} \leftarrow \mathit{Old}(q) \cup \{\eta\}, \\
\qquad\qquad\qquad \mathit{New} \leftarrow \mathit{New}(q) \cup \{p, q\}, \\
\qquad\qquad\qquad \mathit{Next} \leftarrow \mathit{Next}(q)]; \\
\qquad\quad \mathit{expand}(q', \mathit{Nodes}); \\
\quad p \vee q\text{:}\; \ldots \\
\quad p\, \mathcal{U}\, q\text{:}\; \ldots \\
\quad p\, \mathcal{R}\, q\text{:}\; \ldots \\
\quad \bigcirc p\text{:}\; \ldots \\
\textbf{end case}
\end{array}
$$

The Algorithm: Node-Expansion (cont.)

IM NTU

Actions on η (that is not a literal):

- $\eta = p \wedge q$, then both p and q are added to *New*.
- $\eta = p \vee q$, then the node is split, adding p to *New* of one copy, and q to the other.
- $\eta = p\,\mathcal{U}\,q$ ($\cong q \vee (p \wedge \bigcirc(p\,\mathcal{U}\,q))$), then the node is split. For the first copy, p is added to *New* and $p\,\mathcal{U}\,q$ to *Next*. For the other copy, q is added to *New*.
- $\eta = p\,\mathcal{R}\,q$ ($\cong (p \wedge q) \vee (q \wedge \bigcirc(p\,\mathcal{R}\,q))$), similar to $\mathcal{U}$.
- $\eta = \bigcirc p$, then p is added to *Next*.

Note: $p\,\mathcal{W}\,q \cong q \vee (p \wedge \bigcirc(p\,\mathcal{W}\,q))$

The Algorithm: Handling $\mathcal{U}$

case η **of**

$$
\begin{array}{ll}
\ldots & \\
p\,\mathcal{U}\,q: & q_1 := [\mathit{ID} \leftarrow \mathit{new\_ID}(), \\
& \qquad \mathit{Incoming} \leftarrow \mathit{Incoming}(q), \\
& \qquad \mathit{Old} \leftarrow \mathit{Old}(q) \cup \{\eta\}, \\
& \qquad \mathit{New} \leftarrow \mathit{New}(q) \cup \{p\}, \\
& \qquad \mathit{Next} \leftarrow \mathit{Next}(q) \cup \{p\,\mathcal{U}\,q\}]; \\
& q_2 := [\mathit{ID} \leftarrow \mathit{new\_ID}(), \\
& \qquad \mathit{Incoming} \leftarrow \mathit{Incoming}(q), \\
& \qquad \mathit{Old} \leftarrow \mathit{Old}(q) \cup \{\eta\}, \\
& \qquad \mathit{New} \leftarrow \mathit{New}(q) \cup \{q\}, \\
& \qquad \mathit{Next} \leftarrow \mathit{Next}(q)]; \\
& \mathit{expand}(q_2, \mathit{expand}(q_1, \mathit{Nodes})); \\
\ldots &
\end{array}
$$

end case

The Algorithm: Handling $\mathcal{R}$

case η **of**

$\ldots$

$$
\begin{aligned}
p\,\mathcal{R}\,q:\ q_1 &:= [\mathit{ID} \leftarrow \mathit{new\_ID}(),\\
&\quad\ \mathit{Incoming} \leftarrow \mathit{Incoming}(q),\\
&\quad\ \mathit{Old} \leftarrow \mathit{Old}(q) \cup \{\eta\},\\
&\quad\ \mathit{New} \leftarrow \mathit{New}(q) \cup \{q\},\\
&\quad\ \mathit{Next} \leftarrow \mathit{Next}(q) \cup \{p\,\mathcal{R}\,q\}];\\
q_2 &:= [\mathit{ID} \leftarrow \mathit{new\_ID}(),\\
&\quad\ \mathit{Incoming} \leftarrow \mathit{Incoming}(q),\\
&\quad\ \mathit{Old} \leftarrow \mathit{Old}(q) \cup \{\eta\},\\
&\quad\ \mathit{New} \leftarrow \mathit{New}(q) \cup \{p, q\},\\
&\quad\ \mathit{Next} \leftarrow \mathit{Next}(q)];\\
&\mathit{expand}(q_2, \mathit{expand}(q_1, \mathit{Nodes}));
\end{aligned}
$$

$\ldots$

end case

Nodes to GBA

The list of nodes in *Nodes* can now be converted into a generalized Büchi automaton $B = (\Sigma, Q, q_0, \Delta, F)$:

1. Σ consists of sets of propositions from AP.
2. The set of states Q includes the nodes in *Nodes* and the additional initial state q_0.
3. $(r, \alpha, r') \in \Delta$ iff $r \in \mathit{Incoming}(r')$ and α satisfies the conjunction of the negated and nonnegated propositions in $\mathit{Old}(r')$
4. q_0 is the initial state, playing the role of *init*.
5. F contains a separate set F_i of states for each subformula of the form $p \,\mathcal{U}\, q$; F_i contains all the states r such that either $q \in \mathit{Old}(r)$ or $p \,\mathcal{U}\, q \notin \mathit{Old}(r)$.

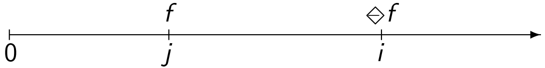
PTL: The Past

- We now add the past fragment.
- Past operators include $\widetilde{\ominus}$ (before), $\ominus$ (previous), $\diamondminus$ (once), $\boxminus$ (so-far), $\mathcal{S}$ (since), and $\mathcal{B}$ (back-to).
- The full PTL formulae are defined inductively as follows:
  - Every variable $p \in V$ is a PTL formula.
  - If f and g are PTL formulae, then so are $\neg f$, $f \vee g$, $f \wedge g$, $\bigcirc f$, $\Diamond f$, $\Box f$, $f\ \mathcal{U}\ g$, $f\ \mathcal{W}\ g$, $\widetilde{\ominus} f$, $\ominus f$, $\diamondminus f$, $\boxminus f$, $f\ \mathcal{S}\ g$, and $f\ \mathcal{B}\ g$.
    ($\neg f \vee g$ is also written as $f \rightarrow g$ and $(f \rightarrow g) \wedge (g \rightarrow f)$ as $f \leftrightarrow g$.)
- Examples:
  - $\Box(p \rightarrow \diamondminus q)$ says "every p is preceded by a q."
  - $\Box(\diamondminus \neg p \rightarrow \diamondminus q)$ is another way of saying $p\ \mathcal{W}\ q$!

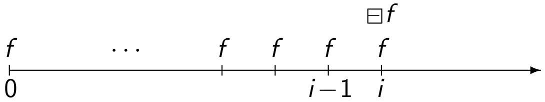
PTL: The Past (cont.)

- For past temporal operators,
  - $(\sigma, i) \models \widetilde{\ominus} f \iff i = 0$ or $(\sigma, i-1) \models f$
  - $(\sigma, i) \models \ominus f \iff i > 0$ and $(\sigma, i-1) \models f$

    The difference between $\widetilde{\ominus} f$ and $\ominus f$ occurs at position 0.
  - $(\sigma, i) \models \diamondminus f \iff$ for some j, $0 \leq j \leq i$, $(\sigma, j) \models f$

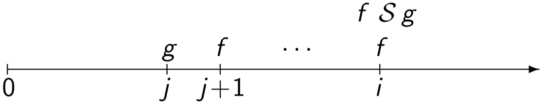


  - $(\sigma, i) \models \boxminus f \iff$ for all j, $0 \leq j \leq i$, $(\sigma, j) \models f$



PTL: The Past (cont.)

- For past temporal operators (cont.),
  - $(\sigma, i) \models f \ \mathcal{S}\, g \iff$ for some j, $0 \leq j \leq i$, $(\sigma, j) \models g$ and for all k, $j < k \leq i$, $(\sigma, k) \models f$



  - $(\sigma, i) \models f \ \mathcal{B}\, g \iff$ (for some j, $0 \leq j \leq i$, $(\sigma, j) \models g$ and for all k, $j < k \leq i$, $(\sigma, k) \models f$) or (for all k, $0 \leq k \leq i$, $(\sigma, k) \models f$)

    $f \ \mathcal{B}\, g$ holds at position i if and only if $f \ \mathcal{S}\, g$ or $\boxminus f$ holds at position i.

A Hierarchy of Temporal Properties

- Classes of temporal properties:
  - Safety properties: $\Box p$
  - Guarantee properties: $\Diamond p$
  - Obligation properties: $\bigwedge_{i=1}^{n}(\Box p_i \vee \Diamond q_i)$
  - Response properties: $\Box\Diamond p$
  - Persistence properties: $\Diamond\Box p$
  - Reactivity properties: $\bigwedge_{i=1}^{n}(\Box\Diamond p_i \vee \Diamond\Box q_i)$

  Here p, q, p_i, q_i are arbitrary past temporal formulae.
- The hierarchy

$$\begin{array}{c} \text{Safety} \\ \text{Guarantee} \end{array} \subseteq \text{Obligation} \subseteq \begin{array}{c} \text{Response} \\ \text{Persistence} \end{array} \subseteq \text{Reactivity}$$

- Every temporal formula is equivalent to some reactivity formula.

More Common Temporal Properties

- Safety properties: $\Box p$
  Example: $p \mathcal{W} q$ is a safety property, as it is equivalent to $\Box(\ominus\neg p \rightarrow \ominus q)$.
- Response properties
  - Canonical form: $\Box\Diamond p$
  - Variant: $\Box(p \rightarrow \Diamond q)$ (p leads-to q), which is equivalent to $\Box\Diamond(\neg p \mathcal{B} q)$.
- Reactivity properties: $\bigwedge_{i=1}^{n}(\Box\Diamond p_i \vee \Diamond\Box q_i)$
- (Simple) reactivity properties
  - Canonical form: $\Box\Diamond p \vee \Diamond\Box q$
  - Variants: $\Box\Diamond p \rightarrow \Box\Diamond q$ or $\Box(\Box\Diamond p \rightarrow \Diamond q)$, which is equivalent to $\Box\Diamond q \vee \Diamond\Box\neg p$.
  - Extended form: $\Box((p \wedge \Box\Diamond r) \rightarrow \Diamond q)$

PTL to Automata: A Tableau Construction

- We next study the Tableau Construction as described in [Manna and Pnueli 1995], which handles both future and past temporal operators.
- More efficient constructions exist, but this construction is relatively easy to understand.
- A tableau is a graphical representation of all models/sequences that satisfy the given temporal logic formula.
- The construction results in essentially a GBA, but leaving propositions on the states (rather than moving them to the incoming edges of a state).
- Our presentation will be slightly different, to make the resulting GBA more apparent.

Expansion Formulae

- The requirement that a temporal formula holds at a position j of a model can often be decomposed into requirements that
  - a simpler formula holds at the same position and
  - some other formula holds either at $j+1$ or $j-1$.
- For this decomposition, we have the following expansion formulae:

$$
\begin{array}{ll}
\Box p \cong p \wedge \bigcirc \Box p & \boxminus p \cong p \wedge \tilde{\ominus} \boxminus p \\
\Diamond p \cong p \vee \bigcirc \Diamond p & \diamondminus p \cong p \vee \ominus \diamondminus p \\
p \,\mathcal{U}\, q \cong q \vee (p \wedge \bigcirc(p \,\mathcal{U}\, q)) & p \,\mathcal{S}\, q \cong q \vee (p \wedge \ominus(p \,\mathcal{S}\, q)) \\
p \,\mathcal{W}\, q \cong q \vee (p \wedge \bigcirc(p \,\mathcal{W}\, q)) & p \,\mathcal{B}\, q \cong q \vee (p \wedge \tilde{\ominus}(p \,\mathcal{B}\, q))
\end{array}
$$

Note: this construction does not deal with $\mathcal{R}$.

Closure

- We define the closure of a formula φ, denoted by Φ_φ, as the smallest set of formulae satisfying the following requirements:
  - $\varphi \in \Phi_\varphi$.
  - For every $p \in \Phi_\varphi$, if q a subformula of p then $q \in \Phi_\varphi$.
  - For every $p \in \Phi_\varphi$, $\neg p \in \Phi_\varphi$.
  - For every $\psi \in \{\Box p, \Diamond p, p\, \mathcal{U}\, q, p\, \mathcal{W}\, q\}$, if $\psi \in \Phi_\varphi$ then $\bigcirc\psi \in \Phi_\varphi$.
  - For every $\psi \in \{ \diamondminus p, p\, \mathcal{S}\, q\}$, if $\psi \in \Phi_\varphi$ then $\ominus\psi \in \Phi_\varphi$.
  - For every $\psi \in \{ \boxminus p, p\, \mathcal{B}\, q\}$, if $\psi \in \Phi_\varphi$ then $\circledtilde\psi \in \Phi_\varphi$.
- So, the closure Φ_φ of a formula φ includes all formulae that are relevant to the truth of φ.

Classification of Formulae

α	$K(\alpha)$
$p \wedge q$	$p,\ q$
$\Box p$	$p,\ \bigcirc\Box p$
$\boxminus p$	$p,\ \circleddash\!\!\sim\ \boxminus p$

β	$K_1(\beta)$	$K_2(\beta)$
$p \vee q$	p	q
$\Diamond p$	p	$\bigcirc\Diamond p$
$\diamondminus p$	p	$\ominus\diamondminus p$
$p\ \mathcal{U}\ q$	q	$p,\ \bigcirc(p\ \mathcal{U}\ q)$
$p\ \mathcal{W}\ q$	q	$p,\ \bigcirc(p\ \mathcal{W}\ q)$
$p\ \mathcal{S}\ q$	q	$p,\ \ominus(p\ \mathcal{S}\ q)$
$p\ \mathcal{B}\ q$	q	$p,\ \circledsim(p\ \mathcal{B}\ q)$

- An α-formula φ holds at position j iff all the $K(\varphi)$-formulae hold at j.
- A β-formula ψ holds at position j iff either $K_1(\psi)$ or all the $K_2(\psi)$-formulae (or both) hold at j.

Atoms

- We define an atom over φ to be a subset $A \subseteq \Phi_\varphi$ satisfying the following requirements:
  - R_{sat} : the conjunction of all state formulae in A is satisfiable.
  - $R_\neg$: for every $p \in \Phi_\varphi$, $p \in A$ iff $\neg p \notin A$.
  - R_α : for every α-formula $p \in \Phi_\varphi$, $p \in A$ iff $K(p) \subseteq A$.
  - R_β : for every β-formula $p \in \Phi_\varphi$, $p \in A$ iff either $K_1(p) \in A$ or $K_2(p) \subseteq A$ (or both).
- For example, if atom A contains the formula $\neg\Diamond p$, it must also contain the formulae $\neg p$ and $\neg\bigcirc\Diamond p$.

Mutually Satisfiable Formulae

- A set of formulae $S \subseteq \Phi_\varphi$ is called mutually satisfiable if there exists a model σ and a position $j \geq 0$, such that every formula $p \in S$ holds at position j of σ.
- The intended meaning of an atom is that it represents a maximal mutually satisfiable set of formulae.

Claim (atoms represent necessary conditions)

Let $S \subseteq \Phi_\varphi$ be a mutually satisfiable set of formulae. Then there exists a φ-atom A such that $S \subseteq A$.

- It is important to realize that inclusion in an atom is only a necessary condition for mutual satisfiability (e.g., $\{\bigcirc p \vee \bigcirc\neg p, \bigcirc p, \bigcirc\neg p, p\}$ is an atom for the formula $\bigcirc p \vee \bigcirc\neg p$).

Basic Formulae

- A formula is called basic if it is either a proposition or has the form $\bigcirc p$, $\ominus p$, or $\circledS p$.
- Basic formulae are important because their presence or absence in an atom uniquely determines all other closure formulae in the same atom.
- Let Φ_φ^+ denote the set of formulae in Φ_φ that are not of the form $\neg\psi$.

Algorithm (atom construction)

1. *Find all basic formulae* $p_1, \cdots, p_b \in \Phi_\varphi^+$.
2. *Construct all* 2^b *combinations.*
3. *Complete each combination into a full atom.*

Example

- Consider the formula $\varphi_1 : \Box p \wedge \Diamond \neg p$ whose basic formulae are

$$p,\ \bigcirc\Box p,\ \bigcirc\Diamond\neg p.$$

- Following is the list of all atoms of φ_1:

$$\begin{array}{lllllll}
A_0: & \{\neg p, & \neg\bigcirc\Box p, & \neg\bigcirc\Diamond\neg p, & \neg\Box p, & \Diamond\neg p, & \neg\varphi_1\} \\
A_1: & \{p, & \neg\bigcirc\Box p, & \neg\bigcirc\Diamond\neg p, & \neg\Box p, & \neg\Diamond\neg p, & \neg\varphi_1\} \\
A_2: & \{\neg p, & \neg\bigcirc\Box p, & \bigcirc\Diamond\neg p, & \neg\Box p, & \Diamond\neg p, & \neg\varphi_1\} \\
A_3: & \{p, & \neg\bigcirc\Box p, & \bigcirc\Diamond\neg p, & \neg\Box p, & \Diamond\neg p, & \neg\varphi_1\} \\
A_4: & \{\neg p, & \bigcirc\Box p, & \neg\bigcirc\Diamond\neg p, & \neg\Box p, & \Diamond\neg p, & \neg\varphi_1\} \\
A_5: & \{p, & \bigcirc\Box p, & \neg\bigcirc\Diamond\neg p, & \Box p, & \neg\Diamond\neg p, & \neg\varphi_1\} \\
A_6: & \{\neg p, & \bigcirc\Box p, & \bigcirc\Diamond\neg p, & \neg\Box p, & \Diamond\neg p, & \neg\varphi_1\} \\
A_7: & \{p, & \bigcirc\Box p, & \bigcirc\Diamond\neg p, & \Box p, & \Diamond\neg p, & \varphi_1\}
\end{array}$$

The Tableau

- Given a formula φ, we construct a directed graph T_φ, called the tableau of φ, by the following algorithm.

Algorithm (tableau construction)

1. *The nodes of T_φ are the atoms of φ.*
2. *Atom A is connected to atom B by a directed edge if all of the following are satisfied:*
   - $R_{\bigcirc}$: *For every $\bigcirc p \in \Phi_\varphi$, $\bigcirc p \in A$ iff $p \in B$.*
   - $R_{\ominus}$: *For every $\ominus p \in \Phi_\varphi$, $p \in A$ iff $\ominus p \in B$.*
   - $R_{\circleddash}$: *For every $\circleddash p \in \Phi_\varphi$, $p \in A$ iff $\circleddash p \in B$.*

- An atom is called initial if it does not contain a formula of the form $\ominus p$ or $\neg \circleddash p$ ($\cong \ominus \neg p$).

Example

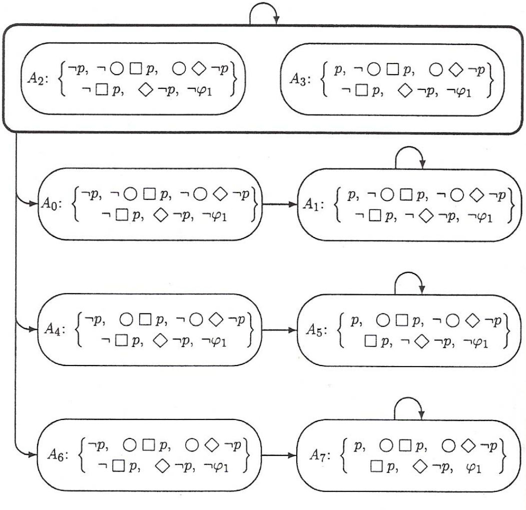


Figure : Tableau T_{φ_1} for $\varphi_1 = \Box p \wedge \Diamond\neg p$. Source: [Manna and Pnueli 1995].

From the Tableau to a GBA

- Create an initial node and link it to every initial atom that contains φ.
- Label each directed edge with the atomic propositions that are contained in the ending atom.
- Add a set of atoms to the accepting set for each subformula of the following form:
  - $\Diamond q$: atoms with q or $\neg\Diamond q$.
  - $p\,\mathcal{U}\,q$: atoms with q or $\neg(p\,\mathcal{U}\,q)$.
  - $\neg\Box\neg q$ ($\cong \Diamond q$): atoms with q or $\Box\neg q$.
  - $\neg(\neg q\,\mathcal{W}\,p)$ ($\cong \neg p\,\mathcal{U}\,(q \wedge \neg p)$): atoms with q or $\neg q\,\mathcal{W}\,p$.
  - $\neg\Box q$ ($\cong \Diamond\neg q$): atoms with $\neg q$ or $\Box q$.
  - $\neg(q\,\mathcal{W}\,p)$ ($\cong \neg p\,\mathcal{U}\,(\neg q \wedge \neg p)$): atoms with $\neg q$ or $q\,\mathcal{W}\,p$.

Correctness: Models vs. Paths

- For a model σ, the infinite atom path $\pi_\sigma : A_0, A_1, \cdots$ in T_φ is said to be induced by σ if, for every position $j \geq 0$ and every closure formula $p \in \Phi_\varphi$,

$$(\sigma, j) \models p \text{ iff } p \in A_j.$$

Claim (models induce paths)

Consider a formula φ and its tableau T_φ. For every model $\sigma : s_0, s_1, \cdots$, there exists an infinite atom path $\pi_\sigma : A_0, A_1, \cdots$ in T_φ induced by σ.

Furthermore, A_0 is an initial atom, and if $\sigma \models \varphi$ then $\varphi \in A_0$.

Correctness: Promising Formulae

- A formula $\psi \in \Phi_\varphi$ is said to promise the formula r if ψ has one of the following forms:

$$\Diamond r,\ \ p\,\mathcal{U}\,r,\ \ \neg\Box\neg r,\ \ \neg(\neg r\,\mathcal{W}\,p).$$

or if r is the negation $\neg q$ and ψ has one of the forms:

$$\neg\Box q,\ \ \neg(q\,\mathcal{W}\,p).$$

Claim (promise fulfillment by models)

Let σ be a model and ψ, a formula promising r. Then, σ contains infinitely many positions $j \geq 0$ such that

$$(\sigma, j) \models \neg\psi \text{ or } (\sigma, j) \models r.$$

Correctness: Fulfilling Paths

- Atom A fulfills a formula ψ that promises r if $\neg\psi \in A$ or $r \in A$.
- A path $\pi : A_0, A_1, \cdots$ in the tableau T_φ is called fulfilling:
  - A_0 is an initial atom.
  - For every promising formula $\psi \in \Phi_\varphi$, π contains infinitely many atoms A_j that fulfill ψ.

Claim (models induce fulfilling paths)

If $\pi_\sigma : A_0, A_1, \cdots$ is a path induced by a model σ, then π_σ is fulfilling.

Correctness: Fulfilling Paths (cont.)

Claim (fulfilling paths induce models)

If $\pi : A_0, A_1, \cdots$ *is a fulfilling path in* T_φ, *there exists a model* σ *inducing* π, *i.e.,* $\pi = \pi_\sigma$ *and, for every* $\psi \in \Phi_\varphi$ *and every* $j \geq 0$,

$$(\sigma, j) \models \psi \textit{ iff } \psi \in A_j.$$

Proposition (satisfiability and fulfilling paths)

Formula φ *is satisfiable iff the tableau* T_φ *contains a fulfilling path* $\pi = A_0, A_1, \cdots$ *such that* A_0 *is an initial* φ*-atom.*

QPTL

IM NTU

- Quantified Propositional Temporal Logic (QPTL) is PTL extended with quantification over boolean variables (so, every PTL formula is also a QPTL formula):
  - If f is a QPTL formula and $x \in V$, then $\forall x : f$ and $\exists x : f$ are QPTL formulae.
- Let $\sigma = s_0 s_1 \cdots$ and $\sigma' = s'_0 s'_1 \cdots$ be two sequences of states.
- We say that σ' is a x-variant of σ if, for every $i \geq 0$, s'_i differs from s_i at most in the valuation of x, i.e., the symmetric set difference of s'_i and s_i is either $\{x\}$ or empty.
- The semantics of QPTL is defined by extending that of PTL with additional semantic definitions for the quantifiers:
  - $(\sigma, i) \models \exists x : f \iff (\sigma', i) \models f$ for some x-variant σ' of σ
  - $(\sigma, i) \models \forall x : f \iff (\sigma', i) \models f$ for all x-variant σ' of σ

Expressiveness

IM NTU

Theorem

PTL is strictly less expressive than Büchi automata.

Proof.

1. Every PTL formula can be translated into an equivalent Büchi automaton.
2. "p holds at every even position" is recognizable by a Büchi automaton, but cannot be expressed in PTL.

□

Theorem

QPTL is expressively equivalent to Büchi automata.

Equivalences and Congruences

- A formula p is valid, denoted $\models p$, if $\sigma \models p$ for every σ.
- Two formulae p and q are equivalent if $\models p \leftrightarrow q$, i.e., $\sigma \models p$ if and only if $\sigma \models q$ for every σ.
- Two formulae p and q are congruent, denoted $p \cong q$, if $\models \Box(p \leftrightarrow q)$.
- Congruence is a stronger relation than equivalence:
  - $p \vee \neg p$ and $\neg \circleddash(p \vee \neg p)$ are equivalent, as they are both true at position 0 of every model.
  - However, they are not congruent; $p \vee \neg p$ holds at all positions of every model, while $\neg \circleddash(p \vee \neg p)$ holds only at position 0.

Congruences

- A minimal set of operators:

$$\neg, \vee, \bigcirc, \ \mathcal{W}, \circledS, \ \mathcal{B}$$

Other operators could be encoded:

$$\begin{array}{ll} & \ominus p \cong \neg \circledS \neg p \\ \Box p \cong p \ \mathcal{W} \ \mathit{False} & \boxminus p \cong p \ \mathcal{B} \ \mathit{False} \\ \Diamond p \cong \neg \Box \neg p & \diamondminus p \cong \neg \boxminus \neg p \\ p \ \mathcal{U} \ q \cong (p \ \mathcal{W} \ q \wedge \Diamond q) & p \ \mathcal{S} \ q \cong (p \ \mathcal{B} \ q \wedge \diamondminus q) \end{array}$$

- Weak vs. strong operators:

$$\begin{array}{ll} \ominus p \cong (\circledS p \wedge \ominus \mathit{True}) & \circledS p \cong (\ominus p \wedge \circledS \mathit{False}) \\ p \ \mathcal{U} \ q \cong (p \ \mathcal{W} \ q \wedge \Diamond q) & p \ \mathcal{W} \ q \cong (p \ \mathcal{U} \ q \vee \Box p) \\ p \ \mathcal{S} \ q \cong (p \ \mathcal{B} \ q \wedge \diamondminus q) & p \ \mathcal{B} \ q \cong (p \ \mathcal{S} \ q \vee \boxminus p) \end{array}$$

Congruences (cont.)

- Duality:

$$
\begin{array}{ll}
\neg\bigcirc p \cong \bigcirc\neg p & \neg\ominus p \cong \widetilde{\ominus}\neg p \\
 & \neg\widetilde{\ominus} p \cong \ominus\neg p \\
\neg\Diamond p \cong \Box\neg p & \neg\diamondminus p \cong \boxminus\neg p \\
\neg\Box p \cong \Diamond\neg p & \neg\boxminus p \cong \diamondminus\neg p \\
\neg(p\ \mathcal{U}\ q) \cong (\neg q)\ \mathcal{W}\ (\neg p \wedge \neg q) & \neg(p\ \mathcal{S}\ q) \cong (\neg q)\ \mathcal{B}\ (\neg p \wedge \neg q) \\
\neg(p\ \mathcal{U}\ q) \cong (\neg p)\ \mathcal{R}\ (\neg q) & \\
\neg(p\ \mathcal{W}\ q) \cong (\neg q)\ \mathcal{U}\ (\neg p \wedge \neg q) & \neg(p\ \mathcal{B}\ q) \cong (\neg q)\ \mathcal{S}\ (\neg p \wedge \neg q) \\
\neg(p\ \mathcal{R}\ q) \cong (\neg p)\ \mathcal{U}\ (\neg q) &
\end{array}
$$

$$
\begin{array}{c}
\neg\exists x : p \cong \forall x : \neg p \\
\neg\forall x : p \cong \exists x : \neg p
\end{array}
$$

- A formula is in the *negation normal form* if negation only occurs in front of an atomic proposition.
- Every PTL/QPTL formula can be converted into an equivalent formula in the negation normal form.

Congruences (cont.)

- Expansion formulae:

$$
\begin{array}{ll}
\Box p \cong p \wedge \bigcirc \Box p & \boxminus p \cong p \wedge \widetilde{\ominus} \boxminus p \\
\Diamond p \cong p \vee \bigcirc \Diamond p & \diamondminus p \cong p \vee \ominus \diamondminus p \\
p \,\mathcal{U}\, q \cong q \vee (p \wedge \bigcirc (p \,\mathcal{U}\, q)) & p \,\mathcal{S}\, q \cong q \vee (p \wedge \ominus (p \,\mathcal{S}\, q)) \\
p \,\mathcal{W}\, q \cong q \vee (p \wedge \bigcirc (p \,\mathcal{W}\, q)) & p \,\mathcal{B}\, q \cong q \vee (p \wedge \widetilde{\ominus} (p \,\mathcal{B}\, q)) \\
p \,\mathcal{R}\, q \cong (q \wedge p) \vee (q \wedge \bigcirc (p \,\mathcal{R}\, q)) &
\end{array}
$$

Note: we have seen that these expansion formulae are essential in translation of a temporal formula into an equivalent Büchi automaton.

Congruences (cont.)

- Idempotence:

$$
\begin{array}{ll}
\Diamond\Diamond p \cong \Diamond p & \diamondminus\diamondminus p \cong \diamondminus p \\
\Box\Box p \cong \Box p & \boxminus\boxminus p \cong \boxminus p \\
p\,\mathcal{U}\,(p\,\mathcal{U}\,q) \cong p\,\mathcal{U}\,q & p\,\mathcal{S}\,(p\,\mathcal{S}\,q) \cong p\,\mathcal{S}\,q \\
p\,\mathcal{W}\,(p\,\mathcal{W}\,q) \cong p\,\mathcal{W}\,q & p\,\mathcal{B}\,(p\,\mathcal{B}\,q) \cong p\,\mathcal{B}\,q \\
(p\,\mathcal{U}\,q)\,\mathcal{U}\,q \cong p\,\mathcal{U}\,q & (p\,\mathcal{S}\,q)\,\mathcal{S}\,q \cong p\,\mathcal{S}\,q \\
(p\,\mathcal{W}\,q)\,\mathcal{W}\,q \cong p\,\mathcal{W}\,q & (p\,\mathcal{B}\,q)\,\mathcal{B}\,q \cong p\,\mathcal{B}\,q
\end{array}
$$

Concluding Remarks

- PTL can be extended in other ways to be as expressive as Büchi automata, i.e., to express all ω-regular properties.
- For example, the industry standard IEEE 1850 Property Specification Language (PSL) is based on an extension that adds classic regular expressions.
- Regarding translation of a temporal formula into an equivalent Büchi automaton, there have been quite a few algorithms proposed in the past.
- How to obtain an automaton as small as possible remains interesting, for both theoretical and practical reasons.

References

- E.M. Clarke, O. Grumberg, and D.A. Peled. *Model Checking*, The MIT Press, 1999.
- E.A. Emerson. Temporal and modal logic, *Handbook of Theoretical Computer Science* (Vol. B), MIT Press, 1990.
- G.J. Holzmann. *The SPIN Model Checker: Primer and Reference Manual*, Addison-Wesley, 2003.
- Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*, Springer, 1992.
- Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems: Safety*, Springer, 1995.