An Alternate View of the Calculation of $E f$ can also be presented by the following more explicit function:

\[
\text{Func}_{E f} (d', D) = \begin{cases} 
0 & \text{if } \text{old} \neq \text{new} \\
\text{old} \text{ while } (\text{old} =: \text{new} \neq 0) \text{ do } \\
(d', D) & \text{else} 
\end{cases}
\]

\[
\text{return } (d, D)
\]

$E f$ states must have a $new$-path leading to a $b$-state

\[
(b \lor new) \land \text{new } \land (d' =: \text{new})
\]

for each $d' \in C(d)$

\[
\text{for each } J \in F do
\]

- Retain states with a $new$-path leading to a $f$-state

\[
(f \lor new) \land new =: \text{new}
\]

- Retain states which have a successor within $new$

\[
\text{new } \lor new =: \text{new}
\]

- Retain states which have a $new$-path leading to a $q$-state

\[
\text{new } \lor \text{new } =: \text{old}
\]

- All retained $p$-states must have a $new$-path leading to a $b$-state

\[
(b \lor new) \land \text{new } \land (d' =: \text{new})
\]

for each $d' \in C(d)$

- for each $J \in F do$

\[
\text{for each } J \in F do
\]

\[
\text{return } (d, D)
\]

\[
\text{return } (d, D)
\]

A. Pnueli

Lecture 7: Model Checking CT1
Synchronous parallel composition is used for the construction of an observer: $O$, which observes and evaluates the behavior of an observed system $D$. Running $D \parallel O$, we let $D$ behave as usual, while $O$ observes its behavior.

The synchronous parallel composition is given by the FDs $D^1, \Theta^1, \Lambda^1$ and $D^2, \Theta^2, \Lambda^2$, denoted by $D^1 \parallel D^2$.

\begin{align*}
\Theta^2 \cup \Theta^1 &= \Theta \\
\Lambda^2 \cup \Lambda^1 &= \Lambda \\
\mathcal{C}^2 \cap \mathcal{C}^1 &= \mathcal{C} \\
\mathcal{L}^2 \cap \mathcal{L}^1 &= \mathcal{L} \\
\mathcal{Z}^2 \cap \mathcal{Z}^1 &= \mathcal{Z}
\end{align*}

where $(\mathcal{C}, \mathcal{Z}, \mathcal{L}, \mathcal{F}, \Theta, \Lambda) = D$ is given by the FDs $D$. The synchronous parallel composition of the systems $D^1$ and $D^2$, denoted by $D^1 \parallel D^2$, is denoted by $D^1 \parallel D^2$. 

**Synchronous Parallel Composition**

A. Pnueli
A basic state formula $\phi$ is called a fair state formula if the single path quantifier appearing in $\phi$ is either $E$ or $A$. Otherwise, it is called an unfair state formula.

### Claim 7. [Sufficient to deal with Fair State Formulas]

For an unfair basic state formula $\phi$ and an FDS $\mathcal{D}$,

$$\phi \models \mathcal{D} \iff \text{fair}\left(\phi\right) \models \mathcal{D}$$

The following claim shows that, with no loss of generality, we can restrict our attention to fair basic state formulas.

The following claim shows that, with no loss of generality, we can restrict our attention to fair basic state formulas.

Let $\mathcal{D}$ be an FDS. We denote by $\mathcal{D}_{\text{unfair}}$ the FDS obtained from $\mathcal{D}$ by removing all justice and compassion requirements.

Obtain $\phi$ by replacing occurrences of $E$ by $E^f$ and occurrences of $A$ by $A^f$.

Given an unfair state formula $\phi$, we denote by $\text{fair}(\phi)$ the formula obtained from $\phi$ by replacing occurrences of $E$ with $E^f$, and occurrences of $A$ with $A^f$. Otherwise, it is called an unfair state formula.
We will start by presenting testers for basic path formulas.

A path formula whose principal operator is temporal, and such that it does not contain any nested temporal operator or path quantifier is called a basic path formula.

\[ \phi = \left[ \ell^I \right] \varphi \quad \text{for every position } \ell, \text{ such that } x \in \ell^I \]

This tester has a distinguished boolean variable called the temporal tester for every LTL formula, \( \phi \).

Temporal Testers
Example: a Test for $\mu$.

Lecture 7: Model Checking CTL

Hardware Verification, NYU, Fall 2004

Example: a Test for $\mu$.

The second justice requirement stipulates that $\mu \gamma$ be false infinitely many times. Note that, since the transition relation implies $\mu \gamma \leftarrow \mu \gamma$, the second justice requirement implies that a computation cannot have even a single state with

Vin $\gamma$.

The justice requirement demands that either $p_1 \in \mathbb{I}$, or $x \in \mathbb{I}$ and $0 = d$ infinitely many times, or $0 = x$ and $0 = d$ infinitely many times. This rules out a computation in which $p_1 = 0$ and $x = 1$ continuously, even though such a state sequence satisfies the requirements of initiality and consecution.

The justice requirement demands that $\mu \gamma \leftarrow \mu \gamma$ infinitely many times. This rules out a computation in which $\mu \gamma = 0$, since the transition relation implies $\mu \gamma \leftarrow \mu \gamma$.

$\mu \gamma$:


\[
\begin{align*}
\emptyset : & \mathcal{C} \\
\mu \gamma \land (x \land d = x) \lor \mu \gamma : & \mathcal{L} \\
\mu \gamma \land \{x : x\} \cup (d \Diamond \mu) \land \mu \gamma : & \Theta \\
\mu \gamma : & \Lambda
\end{align*}
\]

$\Diamond$: Model Checking CTL
The first justification requirements stipulates that either $x = 1$ or $\mathbf{p} = 0$ infinitely many times. This prevents the erroneous solution to the equation $x = \mathbf{p}^x_0$ in which continuously (for all positions) $x = 0$. 

\[
\begin{align*}
\emptyset : \mathcal{C} \\
\mathcal{L} : \mathcal{L} \\
\mathcal{L} \land (x \lor d = x) \lor \mathcal{L} : d \\
\emptyset : \Theta \\
\mathcal{L} : \mathcal{L} \\
\mathcal{L} \land (d = x) \lor \mathcal{L} : d \\
\emptyset : \Theta \\
\{\mathcal{L}, x\} \cap (d)\text{vars} : \Lambda
\end{align*}
\]

\[d \square \mathbf{p} \text{ and } d \bigcirc \mathbf{p} \]
Note the justice requirement by which either either 0 = x or b should hold infinitely many times.
by a path quantifier. Next, we will show how to eliminate a temporal operator which is not preceded for the case that $\phi$ was a basic $\text{CTL}$ formula.

$$||(\exists) f|| = ||(\exists f)||$$

Immediately preceded by a path quantifier, using the reduction in Claim 4, we showed how to get rid of an innermost temporal operator which was

$$I = |L| = |\phi|.$$  

A state formula whose principal operators are a pair $\text{L} \text{O}$ and which does not contain any additional temporal operators or path quantifiers is called a basic $\text{CTL}$ formula.  

$$I = |\phi|.$$  

A basic state formula is a formula of the form $\exists \phi$, where $\phi$ contains no path quantifiers.

$$I = |L|, 0 = |\phi|.$$  

A path formula whose principal operator is temporal, and such that it does not contain any nested temporal operators or path quantifiers is called a basic path formula.

**Reminder: Definitions of Basic Formulas**
elimination of non-CTL temporal operators

Let $Q_f$ be a fair basic state formula containing one or more occurrences of the basic path formula, where $Q_f \in \{ A, E \}$. Then, we can compute the basic path formula, where $\phi$ be a fair basic state formula containing one or more occurrences of a general CTL* formula, as shown by the following:

A similar modularity (though for a higher price) exists for the LTL component of a linear battle.

The modularity of CTL which enabled us to model check a formula by successively computing the modularity of CTL which enabled us to model check a formula by successively computing

$$[\phi]L \parallel A \exists (\phi x) f f x \parallel [\phi]L \parallel A \exists (\phi x) f f x \parallel A [\phi]L \parallel A \exists (\phi x) f f x \parallel A [\phi]L \parallel A \exists (\phi x) f f x \parallel E$$

Claim 8. [Elimination of non-CTL Temporal Operators]

Outline: Model Checking CTL*
Consider the system $D$:

0 = x \text{ while } 1 = d$

The justice requirement is intended to guarantee that we will not have a computation in which continuously $d \leftarrow d \land x$.

First, we construct the tester $\{d_0, x' \downarrow \}$.

We wish to model check the property $A$, so we construct the tester:

$$\forall x \in \mathbb{Z} \land (x \lor d = x) \land \{d_0, x' \downarrow \} : \{d_0 \downarrow \}$$

Example 1/2
Example 2/2

Next, we form the parallel composition $D \parallel E$. Evaluating $A f x$ over $D \parallel E$ over $x$, we obtain $A f x = \top$. We can therefore compute $k A f x; D k = \top \cdot x; E r : 1 = 1$. Concluding that the original FDR satisfies $A f x$.

\[ I = I : \top x A f x = \|A d \top f A\| \]

A. Pnueli

Hardware Verification, NYU, Fall, 2004
A.Pnueli

What Happens When we Run Out of Temporal Operators

Successive elimination of temporal operators may lead us to the situation that we have got no temporal operators left. In such a case we have a formula of the form $\exists p_q$ where $q$ is a path quantifier and $p$ is an assertion.

A good strategy is to stop when we still have a single temporal operator. In such a case we have a CTL formula or a formula which can be transformed into a CTL formula we have a CTL formula. However, for completeness, we also give rules for simplifying a formula of the form

$$I = \|((\forall) x [\square] L \lor (x)[\diamond] L \lor q) \wedge \forall \text{fair} : \exists x, \forall A$$

$$= \|((\forall) x [\diamond] L \lor (x)[\square] L \lor q) \forall A \wedge \forall \text{fair} : \exists x, \forall A$$

$$= \|((x)[\square] L \lor q) x [\diamond] \forall A \wedge \forall \text{fair} : \exists x, \forall A = \|q, \forall [\square] \diamond \forall A$$

A good strategy is to stop when we still have a single temporal operator. In such a case we have a CTL formula. However, for completeness, we also give rules for simplifying a formula of the form

$$\exists d \lor \text{fair} = \|d \exists f$$

$$\exists d \lor \text{fair} = \|d \forall f$$
A Brief History of the Branching vs. Linear Controversy

A. Pnueli

Skilled using Branching Time Temporal Logic.

[AEP81] – Clarke and Emerson, Design and Synthesis of Synchronisation Notation.

Introduced a notation for Branching Time TL.

[EC80] – Emerson and Clarke, Characterizing Correctness Properties of Parallel Programs using CTL Erdos.

Committed to LTL and Established Axiomatisches.


Committed to LTL and Established Axiomatisches.


Distinguished between the Linear and Branching Versions.

[Pan77] – P. The Temporal Logic of Programs (FOCS'77). Did not actually not identify the connection with TL.

[EC80] – Emerson and Clarke, Characterizing Correctness Properties of Parallel Programs using CTL.

A Uniform Logical Basis for the Description, Specification and Verification of Programs. Had some of the ideas of Temporal Logic but did not identify the connection with TL.

F. Kroger, A Uniform Logical Basis for the Description, Specification and Verification of Programs.
Branching Time versus Linear Time.Introduced CTL∗.


[EL85] – Emerson and Lei, Modalities for model checking: Branching time strikes.

[Lam83] – Lamport, ‘Sometimes’ is sometimes ‘Not Never’.

[Lam82] – Lamport, What good is Temporal Logic? Branching Time TL is useless!

[CESAR] – Queille and Sifakis, SpecifiCation and VerifiCation of concurrent SyStems.

[SC85] – Sistla and Clarke, The Complexity of Propositional Linear Temporal Logic. Showed that CTL model checking is polynomial, while LTL model checking is PSPACE-complete.

[LP85] – Lichtenstein and P., Checking that finite state concurrent programs satisfy their linear specification.

In a recent committee Accelera, convened in order to set a standard for a special circuit language for properties of hardware designs, there was a fierce debate between IBM who supported a branching logic, vs. Intel who pushed the linear approach. The committee favored the IBM language.

Currently, the IBM team is considering the incorporation of the linear part as well.

A Recent Debate
This also shows that the past operators add no expressive power, but may add

\[ \forall n \exists t (n \circ t) \]  

Claim 10. Every first-order formula can be translated into a temporal formula in the logic \( L(\forall, \exists) \).

Claim 9. Every first-order formula can be translated into a temporal formula in the logic \( L(\forall, <) \).

W. Kamps [Kamps88] has shown that in our temporal formulas, but then proceeded to show that:

\[
((\forall t) b): t \geq t \in \forall t (t \geq t) \iff (\forall t) d : 0 \geq t + 1
\]

For example, the first-order translation of the formula \( b \circ d \) is

\[
\forall t : (t < t + 1)
\]

Every propositional path formula can be translated into a first-order logic with monadic predicates over the naturals ordered by \( > \) (first-order theory of linear order). Every (propositional) CTL formula has a first-order translation:

**Expressive Completeness**
A formula of the form \( p \) for some past formula \( p \) is called a safety formula.

A formula of the form \( p \) for some past formula \( p \) is called a response formula.

An equivalent characterization is the form \( \bigvee_{k=1}^{?} (b \, \Box \, \Diamond \, q \, b \, \Box) \).

Both formulas state that either there are infinitely many \( q \)'s, or there are no further \( p \)'s, or there is a last \( b \)-position, beyond which there are no further \( d \)'s.

A property is classified as a safety/response property if it can be specified by a safety/response formula.

Every temporal formula is equivalent to a conjunction of a reactivity formula, i.e.

\[ \forall \exists \, \bigwedge (b \, \Box \, \Diamond \, q \, b \, \Box) \]

A property is classified as a 

\( d \) for some past formula \( d \) is called a safety formula.

\( d \) for some past formula \( d \) is called a response formula.

Classifications of Formulas/Properties