The expression $\phi_L \models (x)f \models x$ in the augmented FDS (for the satisfaction of all the variables not in $L$).

Let $f$ be a fair basic state formula containing one or more occurrences of the basic operators.

Claim 5. Elimination of Temporal Operators

A similar modularity (though for a higher price) exists for the $LTL$ component of a general $CTL^*$ formula, as shown by the following:

A unique feature of $CTL^*$, and a major argument in the branching vs. linear battle, is the ability to check a formula by successively computing $\models$ for all of its nested basic formulas, thus, for a long time, been considered to model checking $CTL^*$ which enabled us to model check a formula by successively computing $\models$ for all of its nested basic formulas.
computation in which continuously, we will not have a
The justicement requirement is intended to guarantee that we have a

\[ \emptyset : \mathcal{C} \]
\[ \mathcal{I} : \mathcal{L} \]
\[ x \land \mathcal{I} = x : \mathcal{O} \]
\[ \{x,d\} : \Theta = \Lambda \]

We wish to model check the property $A$. First, we construct the test

Consider the system

Example 1/2

\[ d : 2 \]
\[ d : 1 \]
\[ d : 0 \]
We can therefore conclude that the original PDS satisfies $x \diamond f_A \vdash A$, we obtain $\vdash A$ over $\vdash f_A x$. Evaluating $\vdash f_A x$ gives $\vdash f_A x$.

Next, we form the parallel composition $d \parallel A : \vdash f_A$.

**Example 2/2**
\[
d^\exists \quad \text{fair} \quad = \quad \exists d^f
\]
\[
d^\forall \quad \text{fair} \quad = \quad \forall d^f
\]
\[
d^\exists = \exists d^\exists = \exists d^\forall
\]

However, for completeness, we also give rules for stratiﬁng a formula of the form \( \exists d \): a case we have a \textit{CTL} formula or a formula which can be transformed into a \textit{CTL} formula. In such a case we have a single temporal operator. In such a case we have got none left. In such a case we have a formula of the form \( \exists d \) where \( d \) is a path quantiﬁer and is an assertion. 

Successive elimination of temporal operators may lead us to the situation that we run out of temporal operators.
using Branching Time Temporal Logic.

[CE81] - Clarke and Emerson, Design and Synthesis of Synchronization Skeltons
notation.

programs using fixpoints. Introduced CTL model checking without the temporal

notation for branching time TL.

[EC80] - Emerson and Clarke, Characterizing correctness properties of parallel

established axiomatics.

distinction between the linear and branching versions.

did not actually

identity the connection with TL.


Verificiation of Programs. Had some of the ideas of Temporal Logic but did not

A Brief History of the Branching vs. Linear Controversy

A. Pnueli

Lecture 3: Model Checking CTL
Time versus Linear Time: Introduced CTL.

[EH85] Emerson and Halpern, "Sometimes and Not Never: Revisited: On Branching
back.

[EL85] Emerson and Lei, "Modalities for model checking: Branching time strikes
their linear specification."

[LPS85] Lichtenstein and P., "Checking that finite state concurrent programs satisfy
PSpace-complete.

[SC85] Sistla and Clarke, "The Complexity of Propositional Linear Temporal Logic."

\[ \neg A p \land \Diamond p \]

[Lam86] Lamport, "Sometimes is sometimes, Not Never."

[Lam83] Lamport, "What Good is Temporal Logic. Branching time $TL$ is useless!

CEASAR.

[QF82] Queille and St"{e}rke, "Specification and Verification of Concurrent Systems in

A. Pnueli.
Currently, the IBM team is considering the incorporation of the linear part as well. The committee favored the IBM language. Who supported a brunching logic, vs. Intel who pushed the linear approach. There was a fierce debate between IBM language for properties of hardware designs. In a recent committee ‘Acceler’ convened in order to set a standard for a specification.
convincence and succinctness.

This also shows that the past operators add no expressive power, but may add a

\[ \text{Claim 7. } \forall \text{ first-order formula can be translated into a temporal formula in the} \]

\[ \text{logic } \mathcal{L}(O, \bigcirc, \mu). \]

\[ \text{Claim 8. } \forall \text{ first-order formula can be translated into a temporal formula in the} \]

\[ \text{logic } \mathcal{L}(\mu, \bigcirc, \nu). \]

\[ \text{Kamp [Kamp88] has shown that the answer is negative if we only allow} \]

\[ \text{in our temporal formulas. But then proceed to show that: } \square \]

\[ \forall \text{ Kamp [Kamp88] has shown that the answer is negative if we only allow} \]

\[ \text{can every first-order formula be translated into temporal logic?} \]

\[ (\exists t_1 b) : \exists t_2 \exists z \exists w \leq (\exists t_1 d) : 0 \leq A_{t_1} \]

\[ b \diamond \Leftarrow d \]

For example, the first-order translation of

\[ \text{monadic predicates over the naturals ordered by } (\text{first-order theory of linear order}) \]

\[ \text{Every (propositional) path formula } \phi \text{ can be translated into a first-order logic with} \]

Exhersive Completeness
where $\mathbf{d}$ and $\mathbf{b}$ are past formulas:

$$(\mathbf{b} \square \diamond \land \mathbf{d} \diamond \square) \lor$$

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

safety/response formula.

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

or there is a last $\mathbf{b}$-position, beyond which there are no further $\mathbf{d}$'s

Both formulas state that either there are infinitely many $\mathbf{b}$'s, or there are no $\mathbf{d}$'s,

$$(\mathbf{b} \not\triangleright (\mathbf{d} \triangleright)) \triangleleft \square \sim (\mathbf{b} \triangleleft \mathbf{d}) \square$$

An equivalence characterization is the form $\mathbf{b} \diamond \leftarrow \mathbf{d}$. The equivalence is justified by

A formula of the form $\mathbf{d} \square \diamond$ for some past formula is called a safety formula.

A formula of the form $\mathbf{d} \square \diamond$ for some past formula is called a safety formula.

Classification of Formulas/Properties