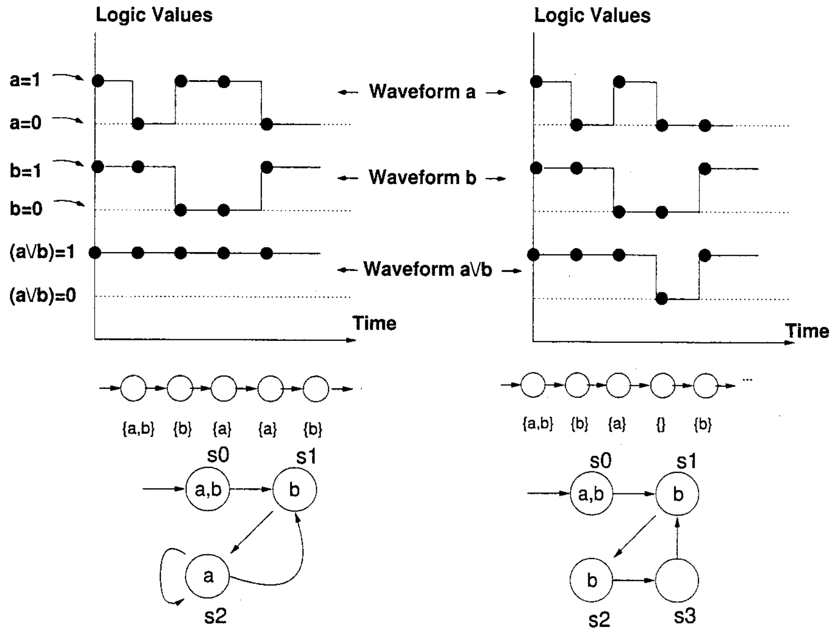
Kapitel 1: Die temporalen Logiken CTL und LTL



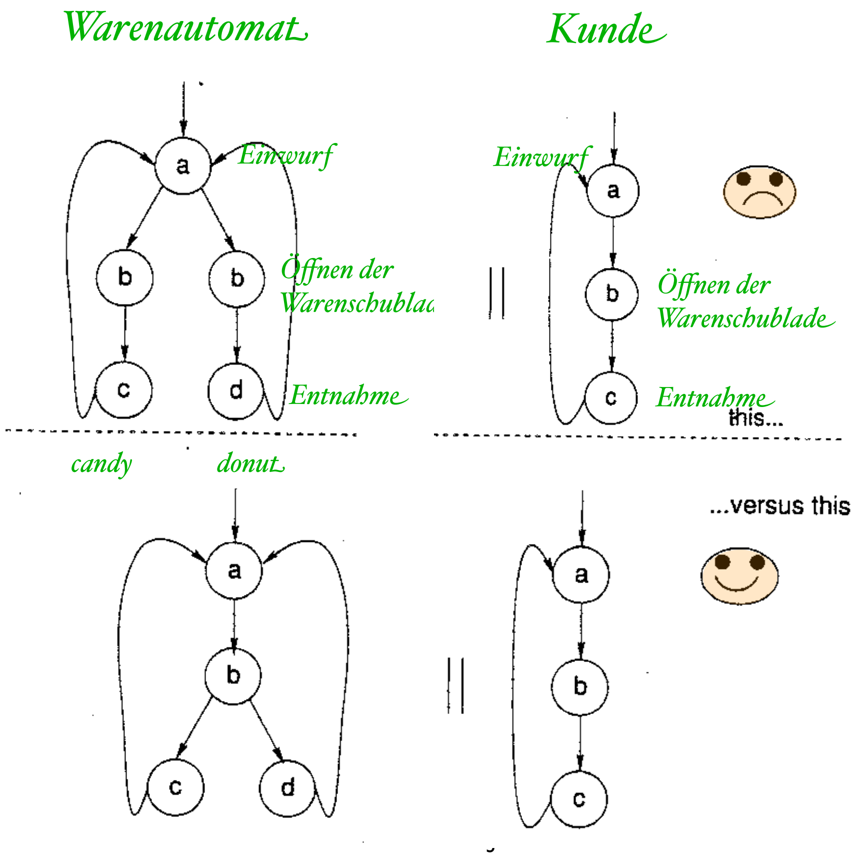
Kripke-Struktur

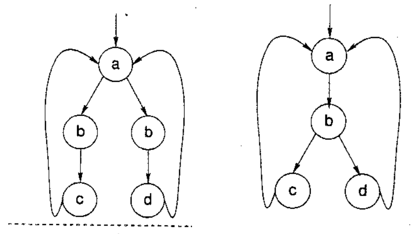
Fig. 22.1. Two Kripke structures and some of their computations. In the Kripke structure on the left, the assertion 'Henceforth $(a \vee b)$' is true.

temporale Logik

Spezialfall der modalen Logik

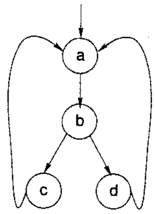
lineare temporale Logik	*verzweigende temporale Logik*
linear temporal logic	*branching temporal logic*
LTL	CTL
Zustandsfolgen	*Berechnungsbäume*
computation sequences	*computation trees*



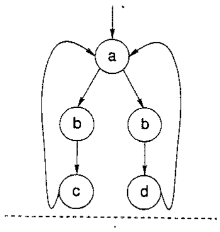


[4] Since *infinite* behaviors of reactive systems are of interest, LTL would view the vending machines as having the set of behaviors described by $(ab(c+d))^\omega$ (see Section 2.8 for an explanation of ω) while CTL would view them in terms of infinite computation trees.

Let us therefore take another approach to see how we can distinguish these machines. In this approach, we unwind the machines into their infinite computation trees, as shown in Figure 22.3 for the Kripke structures of Figure 22.2 and the left-hand side Kripke structure of Figure 22.1. With these computation trees at hand, one can then ask questions such as:



> "At all times, after seeing a b event being offered, does there exist a path leading into a future where the c event is *guaranteed* to be offered?"



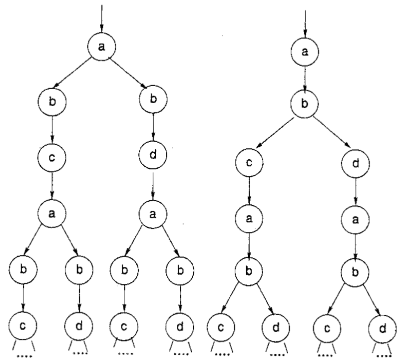


Fig. 22.3. Computation trees for the two Kripke structures of Figure 22.2, as well as the left-hand side Kripke structure of Figure 22.1

CTL formulas are Kripke structure classifiers

Given a CTL formula φ, all possible computation trees fall into two bins—*models* and *non-models*.[5] The computation trees in the *model* ('good') bin are those that satisfy φ while those in the *non-model* ('bad') bin obviously falsify φ.
Consider the CTL formula AG (EF (EG a)) as an example. Here,

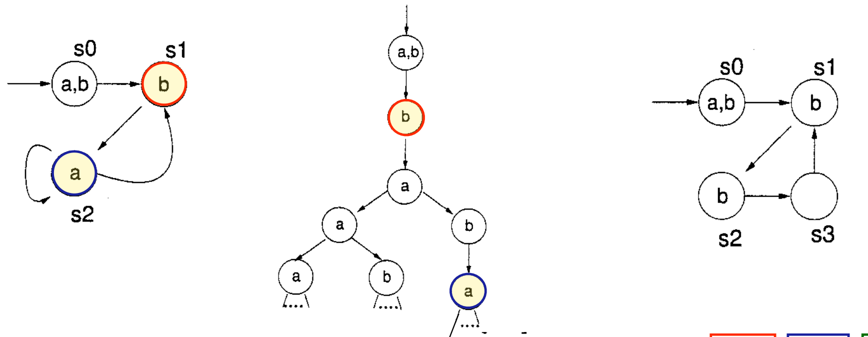
- 'A' is a *path quantifier* and stands for *all paths* at a state
- 'G' is a *state quantifier* and stands for *everywhere along the path*
- 'E' is a *path quantifier* and stands for *exists a path*
- 'F' is a *state quantifier* and stands for *find* (or *future*) along a path
- 'X' is a *state quantifier* and stands for *next* along a path

Zustandsquantoren
Pfadquantoren

The truth of the formula AG (EF (EG a)) can be calculated as follows:

- In all paths, everywhere along those paths, EF (EG a) is true
- The truth of EF (EG a) can be calculated as follows:
  - There exists a path where we will find that EG a is true.
  - The truth of EG a can be calculated as follows:
    - There exists a path where a is globally true.

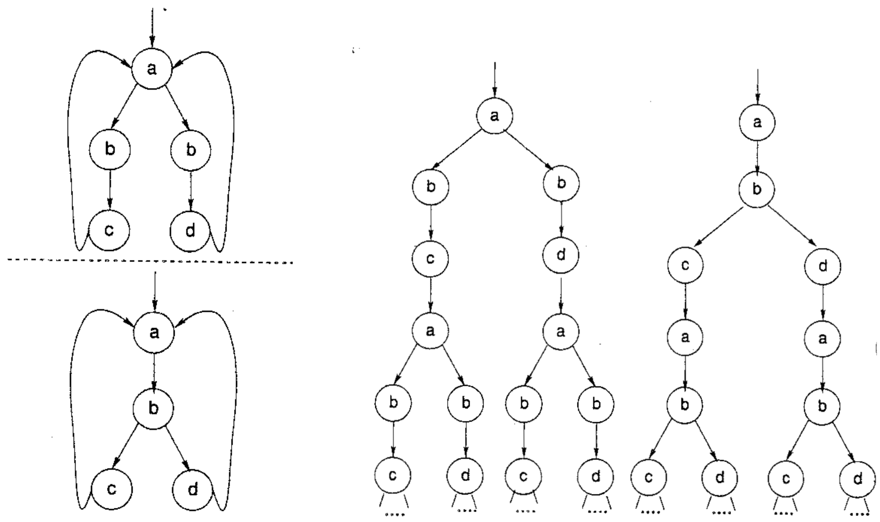
In CTL, one is required to use path quantifiers (A and E) and state quantifiers (G, F, X, and U) in combinations such as AG, AF, AX, AU, EG, EF, EX, and EU.
In other temporal logics such as CTL*, these operators may be used separately; see references such as [20] for details.



Coming back to the examples in Figure 22.1, AG (EF (EG a)) — which means,

> *Starting from any state s of the system (s0, s1, or s2 in our case), one can find a future reachable state t such that starting from t, there is an infinite sequence of states along which a is true*

is true of the Kripke structure on the left, but not the one on the right. This is because wherever we are in the "machine" on the left, we can be permanently stuck in state s2 that emits a. In the machine on the right, a can never be permanent.



the Kripke structures of Figure 22.2, the assertion $AG(b \Rightarrow EX\ c)$ ("wherever we are in the computation, if b is true now, that means that there exists at least one next state where c is true") is true of the bottom Kripke structure but not the top Kripke structure.

LTL *Formeln*

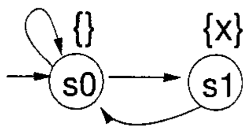
Turning back to LTL, at its core it is a logic of *infinite computations* or *truth-value sequences* ("waveforms"). For example, the LTL formula $\Box(a \vee b)$ (also written as G $(a \vee b)$ or "henceforth $a \vee b$") is true with respect to the computation (waveform) shown on the left-hand side of Figure 22.1 against $(a \vee b) = 1$, while it is false with respect to the waveform shown on the right.

It is customary to view LTL also as a Kripke structure (or computation tree) classifier. This is really a simple extension of the basic idea behind LTL. Under this view, an LTL formula φ is true of a computation tree if and only if *it is true of every infinite path in the tree.* As an example, no LTL formula can distinguish the two Kripke structures given in Figure 22.2, as they both have the same set of infinite paths.

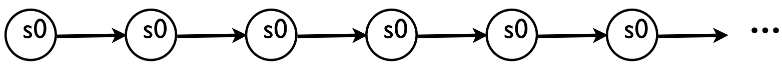
A f

LTL vs. CTL through an example

The differences between LTL and CTL are actually quite subtle. Consider Figure 22.4, and the CTL formula AG (EFx).



AG (EF x) is equivalent to the LTL formula G (F x). ?



22.1.5 LTL syntax

LTL formulas φ are inductively defined as follows, through a context-free grammar:

$\varphi \rightarrow x,$	a propositional variable
$\mid \neg\varphi$	negation of an LTL formula
$\mid (\varphi)$	parenthesization
$\mid \varphi_1 \vee \varphi_2$	disjunction
$\mid \mathrm{G}\varphi$	henceforth φ $\boxed{\square\varphi}$
$\mid \mathrm{F}\varphi$	eventually φ ("future") $\boxed{\Diamond\varphi}$
$\mid \mathrm{X}\varphi$	next φ
$\mid (\varphi_1 \mathrm{U}\, \varphi_2)$	φ_1 until φ_2
$\mid (\varphi_1 \mathrm{W}\, \varphi_2)$	φ_1 weak-until φ_2

X *lokal entspricht* F *global*

$$\sigma = \sigma^0 = s_0, s_1, \ldots,$$
$$\sigma^i = s_i, s_{i+1}, \ldots,$$

$\sigma \models x$	iff x is true at s_0 (written $s_0(x)$)
$\sigma \models \neg\varphi$	iff $\sigma \not\models \varphi$
$\sigma \models (\varphi)$	iff $\sigma \models \varphi$
$\sigma \models \varphi_1 \vee \varphi_2$	iff $\sigma \models \varphi_1 \vee \sigma \models \varphi_2$
$\sigma \models \mathrm{G}\varphi$	iff $\sigma^i \models \varphi$ for every $i \geq 0$
$\sigma \models \mathrm{F}\varphi$	iff $\sigma^i \models \varphi$ for some $i \geq 0$
$\sigma \models \mathrm{X}\varphi$	iff $\sigma^1 \models \varphi$
$\sigma \models (\varphi_1 \mathrm{U}\, \varphi_2)$	iff $\sigma^k \models \varphi_2$ for some $k \geq 0$ and $\sigma^j \models \varphi_1$ for all $j < k$
$\sigma \models (\varphi_1 \mathrm{W}\, \varphi_2)$	iff $\sigma \models \mathrm{G}\varphi_1 \vee \sigma \models (\varphi_1 \mathrm{U}\, \varphi_2)$

LTL example

Consider formula GF x (a common abbreviation for G(F x)). Its semantics are calculated as follows:

$\sigma \models \mathrm{GF}x$ iff $\sigma^i \models \mathrm{F}x$, for all $i \geq 0$
$\sigma^i \models \mathrm{F}x$ iff $\sigma^j \models x$, for some $j \geq i$

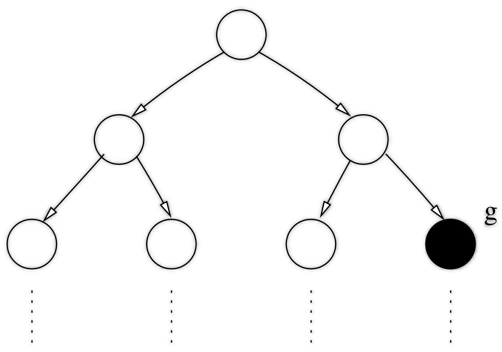
x is true infinitely often

$$\sigma \models (\varphi_1 \mathrm{W}\, \varphi_2) \quad \text{iff } \sigma \models \mathrm{G}\varphi_1 \vee \sigma \models (\varphi_1 \mathrm{U}\, \varphi_2)$$

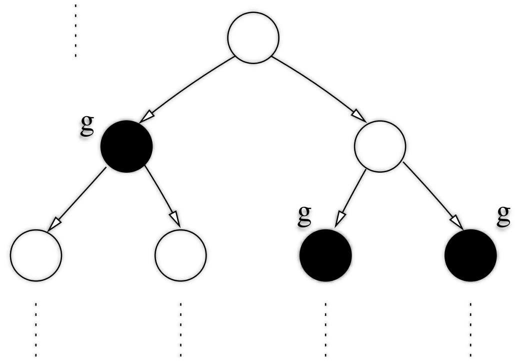
22.1.7 CTL syntax

CTL formulas γ are inductively defined as follows:

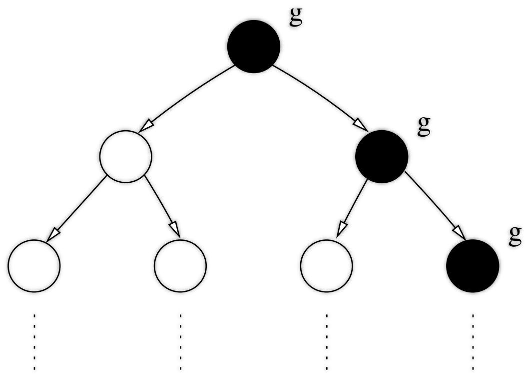
$\gamma \rightarrow x$	a propositional variable	
$\mid \neg\gamma$	negation of γ	
$\mid (\gamma)$	parenthesization of γ	
$\mid \gamma_1 \vee \gamma_2$	disjunction	
$\mid$ AG γ	on all paths,	everywhere along each path
$\mid$ AF γ	on all paths,	somewhere on each path
$\mid$ AX γ	on all paths,	next time on each path
$\mid$ EG γ	on some path,	everywhere on that path
$\mid$ EF γ	on some path,	somewhere on that path
$\mid$ EX γ	on some path,	next time on that path
$\mid$ A[γ_1 U γ_2]	on all paths,	γ_1 until γ_2
$\mid$ E[γ_1 U γ_2]	on some path,	γ_1 until γ_2
$\mid$ A[γ_1 W γ_2]	on all paths,	γ_1 weak-until γ_2
$\mid$ E[γ_1 W γ_2]	on some path,	γ_1 weak-until γ_2



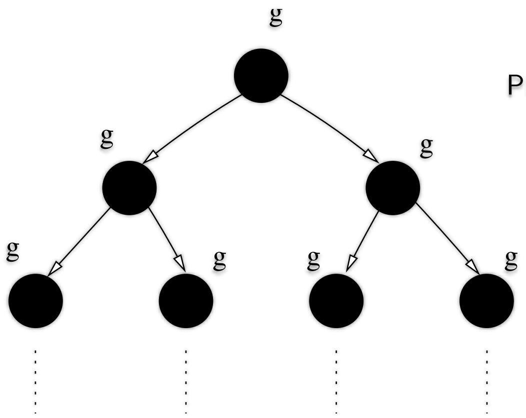
(a) $M, s_0 \models \mathbf{EF}g$



(b) $M, s_0 \models \mathbf{AF}g$



(c) $M, s_0 \models \mathbf{EG}g$



Pl

(d) $M, s_0 \models \mathbf{AG}g$

Beispiele:

- $EF(Start \wedge \neg Ready)$
  Es ist möglich in einen Zustand zu kommen, in dem „*Start*" aber nicht „*Ready*" gilt.

- $AG(Req \rightarrow AF\ Ack)$
  Immer wenn ein Request *Req* erfolgt, dann wird er später einmal mit *Ack* bestätigt.

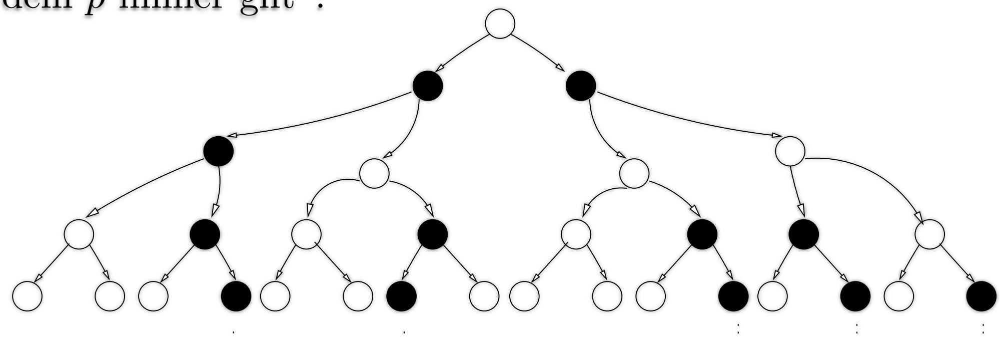
- $AG(AF\ DeviceEnabled)$
  Die Aussage „*DeviceEnabled*" gilt unendlich oft auf jedem Pfad.

- $AG(EF\ Restart)$
  Von jedem Zustand aus ist es möglich, einen Zustand mit „*Restart*" zu erreichen.

Es gibt keine CTL-Formel, die äquivalent zur LTL-Formel

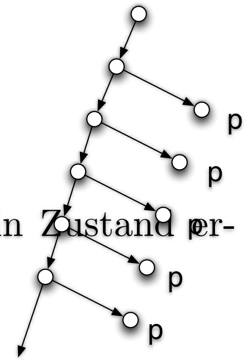
$$A(FGp)$$

ist! Sie bedeutet: „auf jedem Pfad gibt es einen Zustand, ab dem p immer gilt".



Es gibt keine LTL-Formel, die äquivalent zur CTL-Formel:

$$AG(EFp)$$



ist! Sie bedeutet: „Von jedem Zustand ist ein Zustand erreichbar, in dem p gilt."

Ist das äquivalent zu folgender Aussage?

„Alle Pfade enthalten unendlich viele Zustände, in denen p gilt."

$\boxed{AGF\,p}$

Es gibt eine Formel, z.B. $A(FGp) \vee AG(EFp)$, in CTL^*, die weder in CTL noch in LTL ausdrückbar ist. Also: