Linear Clause-Form:

- $x \leftrightarrow (\neg a) \rightsquigarrow \{\neg x,\neg a\},\{x,a\}$
- $x \leftrightarrow (a \wedge b) \rightsquigarrow \{\neg x,a\},\{\neg x,b\},\{x,\neg a,\neg b\}$
- $x \leftrightarrow (a \vee b) \rightsquigarrow \{\neg x,a,b\},\{x,\neg a\},\{x,\neg b\}$
- $x \leftrightarrow (a \oplus b) \rightsquigarrow \{x,\neg a,b\},\{x,\neg a,\neg b\},\{\neg x,a,b\},\{\neg x,\neg a,\neg b\}$
- $x \leftrightarrow (a \rightarrow b) \rightsquigarrow \{\neg x,\neg a,b\},\{x,a\},\{x,\neg b\}$
- $x \leftrightarrow (a \leftrightarrow b) \rightsquigarrow \{x,\neg a,\neg b\},\{\neg x,a,\neg b\},\{\neg x,\neg a,b\},\{x,a,b\}$
- $\rightsquigarrow \{x6\}$ (letzte Klausel nicht vergessen)

FDD: (Nodes von Form (low) $\oplus$ x $\wedge$ (high))

Bsp: $x_1 \oplus x_2$

$= (x_1 \oplus 0) \oplus x_2 \wedge ((x_1 \oplus 0) \oplus (x_1 \oplus 1))$

$= x_1 \oplus x_2 \wedge (x_1 \oplus \neg x_1)$

$= x_1 \oplus x_2 \wedge 1$

$= (0 \oplus x_1 \wedge 1) \oplus x_2 \wedge 1$

$p \oplus q = (p+q)\ \overline{pq} \quad = p\bar{q} + \bar{p}q$

$p \oplus 0 = p \qquad p \oplus 1 = \bar{p}$

$p \leftrightarrow 0 = \bar{p} \qquad p \leftrightarrow 1 = p$

NOT_LEFT_RULE $\dfrac{\neg\phi, \Gamma \vdash \Delta}{\Gamma \vdash \phi, \Delta}$ **NOT_RIGHT_RULE** $\dfrac{\Gamma \vdash \neg\phi, \Delta}{\phi, \Gamma \vdash \Delta}$

AND_LEFT_RULE $\dfrac{\phi \wedge \psi, \Gamma \vdash \Delta}{\phi, \psi, \Gamma \vdash \Delta}$ **AND_RIGHT_RULE** $\dfrac{\Gamma \vdash \phi \wedge \psi, \Delta}{\Gamma \vdash \phi, \Delta \quad \Gamma \vdash \psi, \Delta}$

OR_LEFT_RULE $\dfrac{\phi \vee \psi, \Gamma \vdash \Delta}{\phi, \Gamma \vdash \Delta \quad \psi, \Gamma \vdash \Delta}$ **OR_RIGHT_RULE** $\dfrac{\Gamma \vdash \phi \vee \psi, \Delta}{\Gamma \vdash \phi, \psi, \Delta}$

CUT_RULE $\dfrac{\Gamma \vdash \Delta}{\Gamma \vdash \varphi, \Delta \quad \varphi, \Gamma \vdash \Delta}$

$\mathrm{pre}_\exists^{\mathcal{R}}(\neg q_1 \wedge q_2)$

$= \exists a', q_1', q_2'.\ (q_1' \leftrightarrow (a \oplus q_1))\ \wedge\ (q_2' \leftrightarrow (q_1 \leftrightarrow \neg q_2) \wedge a) \wedge (\neg q_1' \wedge q_2')$

$= (0 \leftrightarrow (a \oplus q_1))\ \wedge\ (1 \leftrightarrow (q_1 \leftrightarrow \neg q_2) \wedge a) \wedge (\neg 0 \wedge 1)$

$= ((q_1 \oplus q_2) \wedge a)\ \wedge\ (a \leftrightarrow q_1)$

$= a \wedge q_1 \wedge \neg q_2$

$\mathrm{suc}_\forall^{\mathcal{R}}(a \vee q_1)$

$= [\forall a, q_1, q_2.\ (q_1' \leftrightarrow (a \oplus q_1))\ \wedge\ (q_2' \leftrightarrow (q_1 \leftrightarrow \neg q_2) \wedge a) \rightarrow (a \vee q_1)]^{a,q_1,q_2}_{a',q_1',q_2'}$

$= [\forall q_2.\ \neg((q_1' \leftrightarrow 0)\ \wedge\ \neg q_2') \vee 0]^{a,q_1,q_2}_{a',q_1',q_2'}$

$= [\neg(\neg q_1'\ \wedge\ \neg q_2')]^{a,q_1,q_2}_{a',q_1',q_2'}$

$= q_1 \vee q_2$

a b	a $\bar{b}$	$\bar{a}$ b	$\bar{a}$ $\bar{b}$	Φ	CNF	DNF	RMNF
0	0	0	0	0	0	0	0
0	0	0	1	$\neg a \wedge \neg b$	$\neg a \wedge (a \vee \neg b)$	$\neg a \wedge \neg b$	$a \wedge b \oplus a \oplus b \oplus 1$
0	0	1	0	$\neg a \wedge b$	$\neg a \wedge (a \vee b)$	$\neg a \wedge b$	$a \wedge b \oplus b$
0	0	1	1	$\neg a$	$\neg a$	$\neg a$	$a \oplus 1$
0	1	0	0	$a \wedge \neg b$	$a \wedge (\neg a \vee \neg b)$	$a \wedge \neg b$	$a \wedge b \oplus a$
0	1	0	1	$\neg b$	$\neg b$	$\neg b$	$b \oplus 1$
0	1	1	0	$a \oplus b$	$(a \vee b) \wedge (\neg a \vee \neg b)$	$a \wedge \neg b \vee \neg a \wedge b$	$a \oplus b$
0	1	1	1	$\neg(a \wedge b)$	$\neg a \vee \neg b$	$\neg a \vee a \wedge \neg b$	$a \wedge b \oplus 1$
1	0	0	0	$a \wedge b$	$a \wedge (\neg a \vee b)$	$a \wedge b$	$a \wedge b$
1	0	0	1	$a \leftrightarrow b$	$(a \vee \neg b) \wedge (\neg a \vee b)$	$\neg a \wedge \neg b \vee a \wedge b$	$a \oplus b \oplus 1$
1	0	1	0	b	b	b	b
1	0	1	1	$a \rightarrow b$	$\neg a \vee b$	$\neg a \vee a \wedge b$	$a \wedge b \oplus a \oplus 1$
1	1	0	0	a	a	a	a
1	1	0	1	$b \rightarrow a$	$a \vee \neg b$	$\neg a \wedge \neg b \vee a$	$a \wedge b \oplus b \oplus 1$
1	1	1	0	$a \vee b$	$a \vee b$	$\neg a \wedge b \vee a$	$a \wedge b \oplus a \oplus b$
1	1	1	1	1	1	1	1

(1) $\dfrac{s \vdash_\Phi \varphi \wedge \psi}{s \vdash_\Phi \varphi \quad s \vdash_\Phi \psi}\ \wedge$ (2) $\dfrac{s \vdash_\Phi \varphi \vee \psi}{s \vdash_\Phi \varphi \quad s \vdash_\Phi \psi}\ \vee$

(3) $\dfrac{s \vdash_\Phi \Box\varphi}{s_1 \vdash_\Phi \varphi\ \ldots\ s_n \vdash_\Phi \varphi}\ \wedge$ (4) $\dfrac{s \vdash_\Phi \Diamond\varphi}{s_1 \vdash_\Phi \varphi\ \ldots\ s_n \vdash_\Phi \varphi}\ \vee$

(5) $\dfrac{s \vdash_\Phi \overleftarrow{\Box}\varphi}{s_1' \vdash_\Phi \varphi\ \ldots\ s_n' \vdash_\Phi \varphi}\ \wedge$ (6) $\dfrac{s \vdash_\Phi \overleftarrow{\Diamond}\varphi}{s_1' \vdash_\Phi \varphi\ \ldots\ s_n' \vdash_\Phi \varphi}\ \vee$

(7) $\dfrac{s \vdash_\Phi \mu x.\varphi}{s \vdash_\Phi \varphi}$ (8) $\dfrac{s \vdash_\Phi \nu x.\varphi}{s \vdash_\Phi \varphi}$

(9) $\dfrac{s \vdash_\Phi x}{s \vdash_\Phi \mathfrak{D}_\Phi(x)}$

in (3)-(6) we assume that $\{s_1, \ldots, s_n\} = \mathrm{suc}_\exists^{\mathcal{R}}(\{s\})$ and $\{s_1', \ldots, s_n'\} = \mathrm{pre}_\exists^{\mathcal{R}}(\{s\})$

```
function Compose(int x, BddNode φ, α)
  int m; BddNode h, l;
  if x > label(φ) then return φ;
  elsif x = label(φ) then
    return ITE(α, high(φ), low(φ));
  else
    m = max{label(φ), label(α)};
    (α0, α1) := Ops(α, m);
    (φ0, φ1) := Ops(φ, m);
    h := Compose(x, φ1, α1);
    l := Compose(x, φ0, α0);
    return CreateNode(m, h, l);
  endif;
end function
```

```
function Constrain(Φ, β)
  if β = 0 then return 0
  elsif Φ ∈ {0,1} ∨ (β = 1) then return Φ
  else
    m = max{label(β), label(Φ)};
    (Φ0, Φ1) := Ops(Φ, m);
    (β0, β1) := Ops(β, m);
    if β0 = 0 then return Constrain(Φ1, β1)
    elsif β1 = 0 then return Constrain(Φ0, β0)
    else l := Constrain(Φ0, β0);
         h := Constrain(Φ1, β1);
         return CreateNode(m, h, l);
    endif
end
```

```
function Restrict(Φ, β)
  if β = 0 then return 0
  elsif Φ ∈ {0,1} ∨ (β = 1) then return Φ
  else
    m = max{label(β), label(Φ)};
    (Φ0, Φ1) := Ops(Φ, m);
    (β0, β1) := Ops(β, m);
    if β0 = 0 then return Restrict(Φ1, β1)
    elsif β1 = 0 then return Restrict(Φ0, β0)
    elsif m = label(Φ) then
      return CreateNode(m, Restrict(Φ1, β1), Restrict(Φ0, β0));
    else
      return Restrict(Φ, Apply(∨, β0, β1))
    endif endif
end
```

```
function Ops(v, m)
  x := label(v);
  if m = degree(x) then
    return (low(v), high(v))
  else return (v, v)
  end
end
```

```
function Apply(⊙, BddNode a, b)
  int m; BddNode h, l;
  if isLeaf(a) & isLeaf(b) then
    return Eval(⊙, label(a), label(b));
  else
    m = max{label(a), label(b)};
    (a0, a1) := Ops(a, m);
    (b0, b1) := Ops(b, m);
    h := Apply(⊙, a1, b1);
    l := Apply(⊙, a0, b0);
    return CreateNode(m, h, l);
  end;
end
```

```
function ITE(BddNode i, j, k)
  int m; BddNode h, l;
  if i = 0 then return k
  elsif i = 1 then return j
  elsif j = k then return k
  else
    m = max{label(i), label(j), label(k)};
    (i0, i1) := Ops(i, m);
    (j0, j1) := Ops(j, m);
    (k0, k1) := Ops(k, m);
    l := ITE(i0, j0, k0);
    h := ITE(i1, j1, k1);
    return CreateNode(m, h, l)
  end
end
```

some CTL* formulas can however be easily translated to CTL

for example, EFGφ = EFEGφ

this principle can be extended (for state formulas ψ):

EXφ = EXEφ	AXφ = AXAφ
EFφ = EFEφ	AGφ = AGAφ
E[φ W ψ] = E[(Eφ) W ψ]	A[φ W ψ] = A[(Aφ) W ψ]
E[φ <u>W</u> ψ] = E[(Eφ) <u>W</u> ψ]	A[φ <u>W</u> ψ] = A[(Aφ) <u>W</u> ψ]
E[ψ U φ] = E[ψ U (Eφ)]	A[φ U ψ] = A[(Aφ) U ψ]
E[ψ <u>U</u> φ] = E[ψ <u>U</u> (Eφ)]	A[φ <u>U</u> ψ] = A[(Aφ) <u>U</u> ψ]
E[φ B ψ] = E[(Eφ) B ψ]	A[ψ B φ] = A[ψ B (Eφ)]
E[φ <u>B</u> ψ] = E[(Eφ) <u>B</u> ψ]	A[ψ <u>B</u> φ] = A[ψ <u>B</u> (Eφ)]

```
function Exists(BddNode e, φ)
  int m; BddNode h, l;
  if isLeaf(φ) ∨ isLeaf(e) then return φ;
  elsif label(e) > label(φ) then (* label(e) does not occur in φ *)
    return Exists(high(e), φ);
  elsif label(e) = label(φ) then
    h := Exists(high(e), high(φ));
    l := Exists(high(e), low(φ));
    return Apply(∨, l, h);
  else (* label(e) < label(φ) *)
    h := Exists(e, high(φ));
    l := Exists(e, low(φ));
    return CreateNode(label(φ), h, l);
  endif;
end function
```

- $[\![\nu x.\varphi \wedge \Diamond x]\!]_\mathcal{K}$ (safety properties)
  - contains states s where an infinite path π starts with $\forall t.\ \pi^{(t)} \in [\![\varphi]\!]_\mathcal{K}$
  - $\rightsquigarrow$ φ holds always on π
- $[\![\mu x.\varphi \vee \Diamond x]\!]_\mathcal{K}$ (liveness properties)
  - contains states s where a (possibly finite) path π starts with $\exists t.\ \pi^{(t)} \in [\![\varphi]\!]_\mathcal{K}$
  - $\rightsquigarrow$ φ holds at least once on π
- $[\![\mu y.[\nu x.\varphi \wedge \Diamond x] \vee \Diamond y]\!]_\mathcal{K}$ (persistence properties)
  - contains states s where an infinite path π starts with $\exists t_1.\forall t_2.\ \pi^{(t_1+t_2)} \in [\![\varphi]\!]_\mathcal{K}$
  - $\rightsquigarrow$ φ holds after some point on π
- $[\![\nu y.\Diamond[\mu x.(y \wedge \varphi) \vee \Diamond x]]\!]_\mathcal{K}$ (fairness properties)
  - contains states s where an infinite path π starts with $\forall t_1.\exists t_2.\ \pi^{(t_1+t_2)} \in [\![\varphi]\!]_\mathcal{K}$
  - $\rightsquigarrow$ φ holds infinitely often on π

- we considered LTL with X, $\overleftarrow{\mathsf{X}}$, [<u>U</u>] and [$\overleftarrow{\underline{\mathsf{U}}}$]
- here is some syntactic sugar
  - Gφ = ¬[1 <u>U</u> (¬φ)]
  - Fφ = [1 <u>U</u> φ]
  - [φ W ψ] = ¬[(¬φ ∨ ¬ψ) <u>U</u> (¬φ ∧ ψ)]
  - [φ <u>W</u> ψ] = [(¬ψ) <u>U</u> (φ ∧ ψ)]
  - [φ B ψ] = ¬[(¬φ) <u>U</u> ψ]
  - [φ <u>B</u> ψ] = [(¬ψ) <u>U</u> (φ ∧ ¬ψ)]
  - [φ U ψ] = ¬[(¬ψ) <u>U</u> (¬φ ∧ ¬ψ)]
- analog definitions hold for past operators
- there is also syntactic sugar for CTL:
- **existential operators**
  - EFφ = E[1 <u>U</u> φ]
  - E[φ U ψ] = E[φ <u>U</u> ψ] ∨ EGφ
  - E[φ <u>B</u> ψ] = E[(¬ψ) <u>U</u> (φ ∧ ¬ψ)]
  - E[φ <u>W</u> ψ] = E[(¬ψ) <u>U</u> (φ ∧ ψ)]
  - E[φ B ψ] = E[(¬ψ) <u>U</u> (φ ∧ ¬ψ)] ∨ EG¬ψ
  - E[φ W ψ] = E[(¬ψ) <u>U</u> (φ ∧ ψ)] ∨ EG¬ψ
- conversely, all binary operators can express [<u>U</u>]
  - [φ <u>U</u> ψ] = ¬[(¬ψ) U (¬φ ∧ ¬ψ)]
  - [φ <u>U</u> ψ] = ¬[(¬ψ) W (φ → ψ)]
  - [φ <u>U</u> ψ] = [ψ <u>W</u> (φ → ψ)]
  - [φ <u>U</u> ψ] = ¬[(¬φ) B ψ]
  - [φ <u>U</u> ψ] = [ψ <u>B</u> (¬φ ∧ ¬ψ)]
- analog definitions hold for past operators
- there is also syntactic sugar for CTL:
- **universal operators**
  - AXφ = ¬EX¬φ
  - AGφ = ¬E[1 <u>U</u> ¬φ]
  - AFφ = ¬EG¬φ
  - A[φ U ψ] = ¬E[(¬ψ) <u>U</u> (¬φ ∧ ¬ψ)]
  - A[φ B ψ] = ¬E[(¬φ) <u>U</u> ψ]
  - A[φ W ψ] = ¬E[(¬ψ) <u>U</u> (¬φ ∧ ψ)]
  - A[φ <u>U</u> ψ] = ¬E[(¬ψ) <u>U</u> (¬φ ∧ ¬ψ)] ∧ ¬EG¬ψ
  - A[φ <u>B</u> ψ] = ¬E[(¬φ ∨ ψ) <u>U</u> ψ] ∧ ¬EG(¬φ ∨ ψ)
  - A[φ <u>W</u> ψ] = ¬E[(¬ψ) <u>U</u> (¬φ ∧ ψ)] ∧ ¬EG¬ψ

- Boolean operations of G
  - ¬Gφ = F¬φ
  - Gφ ∧ Gψ = G[φ ∧ ψ]
  - Gφ ∨ Gψ = $\mathcal{A}_\exists(\{p,q\}, p \wedge q, [p' \leftrightarrow p \wedge \varphi] \wedge [q' \leftrightarrow q \wedge \psi], \mathsf{G}[p \vee q])$
- Boolean operations of F
  - ¬Fφ = G¬φ
  - Fφ ∨ Fψ = F[φ ∨ ψ]
  - Fφ ∧ Fψ = $\mathcal{A}_\exists(\{p,q\}, \neg p \wedge \neg q, [p' \leftrightarrow p \vee \varphi] \wedge [q' \leftrightarrow q \vee \psi], \mathsf{F}[p \wedge q])$

$\rightsquigarrow$ G/F conditions are closed under ∧ and ∨, but not under ¬

- $[\![\nu x.\varphi \wedge \Diamond x]\!]_\mathcal{K}$ (safety properties)
  - contains states s where an infinite path π starts with $\forall t.\ \pi^{(t)} \in [\![\varphi]\!]_\mathcal{K}$
  - $\rightsquigarrow$ φ holds always on π
- $[\![\mu x.\varphi \vee \Diamond x]\!]_\mathcal{K}$ (liveness properties)
  - contains states s where a (possibly finite) path π starts with $\exists t.\ \pi^{(t)} \in [\![\varphi]\!]_\mathcal{K}$
  - $\rightsquigarrow$ φ holds at least once on π
- $[\![\mu y.[\nu x.\varphi \wedge \Diamond x] \vee \Diamond y]\!]_\mathcal{K}$ (persistence properties)
  - contains states s where an infinite path π starts with $\exists t_1.\forall t_2.\ \pi^{(t_1+t_2)} \in [\![\varphi]\!]_\mathcal{K}$
  - $\rightsquigarrow$ φ holds after some point on π
- $[\![\nu y.\Diamond[\mu x.(y \wedge \varphi) \vee \Diamond x]]\!]_\mathcal{K}$ (fairness properties)
  - contains states s where an infinite path π starts with $\forall t_1.\exists t_2.\ \pi^{(t_1+t_2)} \in [\![\varphi]\!]_\mathcal{K}$
  - $\rightsquigarrow$ φ holds infinitely often on π

- $\mathrm{pre}_\exists^{\mathcal{R}}(Q_2) := \{s_1 \in \mathcal{S}_1 \mid \exists s_2.(s_1,s_2) \in \mathcal{R} \wedge s_2 \in Q_2\}$
- $\mathrm{pre}_\forall^{\mathcal{R}}(Q_2) := \{s_1 \in \mathcal{S}_1 \mid \forall s_2.(s_1,s_2) \in \mathcal{R} \rightarrow s_2 \in Q_2\}$
- $\mathrm{suc}_\exists^{\mathcal{R}}(Q_1) := \{s_2 \in \mathcal{S}_2 \mid \exists s_1.(s_1,s_2) \in \mathcal{R} \wedge s_1 \in Q_1\}$
- $\mathrm{suc}_\forall^{\mathcal{R}}(Q_1) := \{s_2 \in \mathcal{S}_2 \mid \forall s_1.(s_1,s_2) \in \mathcal{R} \rightarrow s_1 \in Q_1\}$
- for structures
  - $\mathrm{pre}_\exists^{\mathcal{R}}(Q_2)$ is the set of states that have a successor in Q_2
  - $\mathrm{pre}_\forall^{\mathcal{R}}(Q_2)$ is the set of states that have no successor in $\mathcal{S} \setminus Q_2$

- **CTL can be easily translated to μ-calculus**
- in fact, CTL operators may be viewed as macros
  - AX$\varphi = \Box(\Phi_{\mathrm{inf}} \rightarrow \varphi)$
  - AG$\varphi = \nu x.\ (\Phi_{\mathrm{inf}} \rightarrow \varphi) \wedge \Box x$
  - AF$\varphi = \mu x.\ \varphi \vee \Box x$
  - A[φ <u>U</u> ψ] $= \mu x.\ \psi \vee (\Phi_{\mathrm{inf}} \rightarrow \varphi) \wedge \Box x$
  - A[φ U ψ] $= \nu x.\ \psi \vee (\Phi_{\mathrm{inf}} \rightarrow \varphi) \wedge \Box x$
  - A[φ <u>B</u> ψ] $= \mu x.\ (\Phi_{\mathrm{inf}} \rightarrow \neg\psi) \wedge (\varphi \vee \Box x)$
  - A[φ B ψ] $= \nu x.\ (\Phi_{\mathrm{inf}} \rightarrow \neg\psi) \wedge (\varphi \vee \Box x)$
  - EX$\varphi = \Diamond(\Phi_{\mathrm{inf}} \wedge \varphi)$
  - EG$\varphi = \nu x.\ \varphi \wedge \Diamond x$
  - EF$\varphi = \mu x.\ \Phi_{\mathrm{inf}} \wedge \varphi \vee \Diamond x$
  - E[φ <u>U</u> ψ] $= \mu x.\ (\Phi_{\mathrm{inf}} \wedge \psi) \vee \varphi \wedge \Diamond x$
  - E[φ U ψ] $= \nu x.\ (\Phi_{\mathrm{inf}} \wedge \psi) \vee \varphi \wedge \Diamond x$
  - E[φ <u>B</u> ψ] $= \mu x.\ \neg\psi \wedge (\Phi_{\mathrm{inf}} \wedge \varphi \vee \Diamond x)$
  - E[φ B ψ] $= \nu x.\ \neg\psi \wedge (\Phi_{\mathrm{inf}} \wedge \varphi \vee \Diamond x)$
- using $\Phi_{\mathrm{inf}} = \nu y.\Diamond y$, i.e., the states that have an infi

- Boolean operations of FG
  - ¬FGφ = GF¬φ
  - FGφ ∧ FGψ = FG[φ ∧ ψ]
  - FGφ ∨ FGψ = $\mathcal{A}_\exists(\{q\}, \neg q, q' \leftrightarrow (q \Rightarrow \psi \mid \neg\varphi), \mathsf{FG}[\neg q \vee \psi])$
- Boolean operations of GF
  - ¬GFφ = FG¬φ
  - GFφ ∨ GFψ = GF[φ ∨ ψ]
  - GFφ ∧ GFψ = $\mathcal{A}_\exists(\{q\}, \neg q, q' \leftrightarrow (q \Rightarrow \neg\psi \mid \varphi), \mathsf{GF}[q \wedge \psi])$

$\rightsquigarrow$ GF/FG conditions are closed under ∧ and ∨, but not under ¬

- reduction of G
  - Gφ = $\mathcal{A}_\exists(\{q\}, q, \varphi \wedge q \wedge q', \mathsf{F}q)$
  - Gφ = $\mathcal{A}_\exists(\{q\}, q, q' \leftrightarrow q \wedge \varphi, \mathsf{FG}q)$
  - Gφ = $\mathcal{A}_\exists(\{q\}, q, q' \leftrightarrow q \wedge \varphi, \mathsf{GF}q)$
- reduction of F
  - Fφ can not be expressed by NDet_G
  - Fφ = $\mathcal{A}_\exists(\{q\}, \neg q, q' \leftrightarrow q \vee \varphi, \mathsf{FG}q)$
  - Fφ = $\mathcal{A}_\exists(\{q\}, \neg q, q' \leftrightarrow q \vee \varphi, \mathsf{GF}q)$
- again, watchdogs are employed
- **interestingly, Gφ can be expressed by NDet_F!**

- using EX, EG, and E[<u>U</u>] as basic operators requires two temporal operators to define the previous syntactic sugar
- better alternative: basic operators EX, E[U], and E[<u>U</u>]:
- **existential operators**
  - EGφ = E[φ U 0]
  - EFφ = E[1 <u>U</u> φ]
  - E[φ B ψ] = E[(¬ψ) U (φ ∧ ¬ψ)]
  - E[φ <u>B</u> ψ] = E[(¬ψ) <u>U</u> (φ ∧ ¬ψ)]
  - E[φ W ψ] = E[(¬ψ) U (φ ∧ ψ)]
  - E[φ <u>W</u> ψ] = E[(¬ψ) <u>U</u> (φ ∧ ψ)]
- **universal operators**
  - AXφ = ¬EX¬φ
  - AGφ = ¬E[1 <u>U</u> ¬φ]
  - AFφ = ¬E[(¬φ) U 0]
  - A[φ U ψ] = ¬E[(¬ψ) <u>U</u> (¬φ ∧ ¬ψ)]
  - A[φ <u>U</u> ψ] = ¬E[(¬ψ) U (¬φ ∧ ¬ψ)]
  - A[φ B ψ] = ¬E[(¬φ) <u>U</u> ψ]
  - A[φ <u>B</u> ψ] = ¬E[(¬φ) U ψ]
  - A[φ W ψ] = ¬E[(¬ψ) <u>U</u> (¬φ ∧ ψ)]
  - A[φ <u>W</u> ψ] = ¬E[(¬ψ) U (¬φ ∧ ψ)]