# Tinlab advanced algorithms

W. Oele

8 maart 2020

# Deze les

- Computation trees
- Computation tree logic (ctl)
- fairness, freedom of starvation
- liveness

# Logica herhaling

Wetten van De Morgan:

- $(p \wedge q) \equiv \neg(\neg p \vee \neg q)$
- $(p \vee q) \equiv \neg(\neg p \wedge \neg q)$

# Logica herhaling

Wetten van De Morgan:

- $(p \wedge q) \equiv \neg(\neg p \vee \neg q)$
- $(p \vee q) \equiv \neg(\neg p \wedge \neg q)$

Kwantoren:

- $\forall x P(x)$: *"Voor alle x geldt P(x)."*

# Logica herhaling

Wetten van De Morgan:

- $(p \wedge q) \equiv \neg(\neg p \vee \neg q)$
- $(p \vee q) \equiv \neg(\neg p \wedge \neg q)$

Kwantoren:

- $\forall x P(x)$: *"Voor alle x geldt P(x)."*
- $\exists x P(x)$: *"Er is een x, waarvoor geldt: P(x).*

# Logica herhaling

Wetten van De Morgan:

- $(p \wedge q) \equiv \neg(\neg p \vee \neg q)$
- $(p \vee q) \equiv \neg(\neg p \wedge \neg q)$

Kwantoren:

- $\forall x P(x)$: *"Voor alle $x$ geldt $P(x)$."*
- $\exists x P(x)$: *"Er is een $x$, waarvoor geldt: $P(x)$.*

Wanneer men de variabelen in een predicaat bindt aan een specifieke waarde, verkrijgen we een propositie.

# Predicaten logica: dualiteiten

Kwantoren:

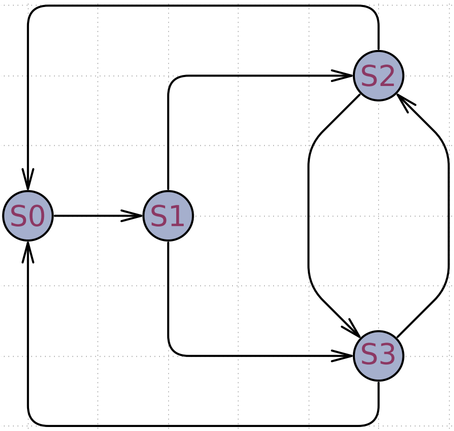
- $\neg\forall x P(x) \equiv \exists x \neg P(x)$
- $\neg\exists x P(x) \equiv \forall x \neg P(x)$

# De Kripke structuur

Een Kripke structuur bestaat uit een 4-tuple $M = (S, S_0, R, L)$ met daarin:

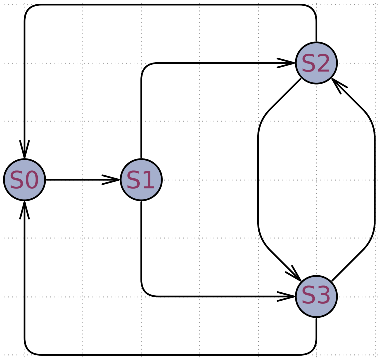
- $S$: de verzameling van alle states van dit systeem.
- $S_0 \subseteq S$: de verzameling van beginstates.
- $R \subseteq S \times S$: de transitierelatie.
- $L : s \to 2^{AP}$: labels, waarmee we iedere state labelen met atomaire proposities die waar zijn in die state.

# Een systeem. . .

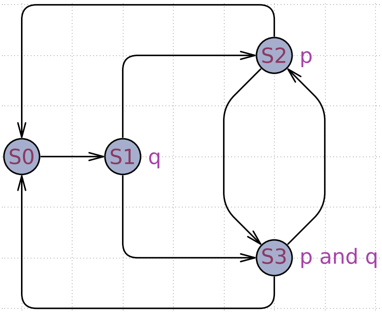


# Een systeem. . .

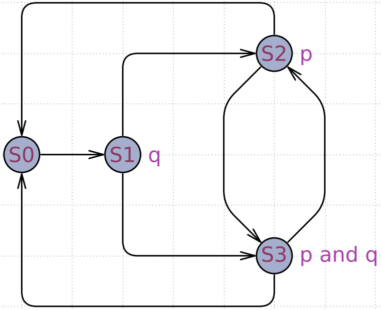
Wens: uitspraken zoals $p$ en $q$ kunnen doen over dit systeem. . .



# Een systeem. . .



# Een systeem. . .




Probleem: Uitspraken $p$ en $q$ in de ene state wel waar, maar in de andere state niet. . .

# Systemen en propositielogica

Probleem:

- Proposities $p$ en $q$ in de ene state wel waar, maar in de andere state niet. . .
- Propositielogica "niet sterk" genoeg om uitspraken te kunnen doen over systemen/werelden, waarin voortdurend dingen veranderen.

# Systemen en propositielogica

Probleem:

- Proposities $p$ en $q$ in de ene state wel waar, maar in de andere state niet...
- Propositielogica "niet sterk" genoeg om uitspraken te kunnen doen over systemen/werelden, waarin voortdurend dingen veranderen.

Uitdaging:

- Een logica ontwikkelen die wel met veranderingen kan omgaan...

# Computation tree logic (ctl)

- Ctl is een vorm van *temporele* logica.
- Er zijn vele temporele logica' s:
  - Computation tree logic
  - Linear tree logic
  - CTL*: combinatie van bovenstaande twee
  - ...

De tool Uppaal "begrijpt" een subset van ctl.

# Computation tree logic (ctl)

Ctl:

- maakt het mogelijk logische uitspraken te doen over de *computation tree* van een Kripke structuur.
- De vraag of een Kripke structuur een bepaalde in ctl geformuleerde eigenschap bezit kan middels een algoritme worden beantwoord.
- Daarmee is het volledig geautomatiseerd checken van een (model van een) systeem mogelijk.

# Computation tree logic: Semantiek

Gegeven een Kripke structuur:

$$M = (S, S_0, R, L)$$

# Computation tree logic: Semantiek

Gegeven een Kripke structuur:

$$M = (S, S_0, R, L)$$

Formule $\phi$ is waar in state $s$ van de Kripke structuur $M$
*"Formula $\phi$ holds in state $s$ of Kripke structure $M$"*:

$$M, s \models \phi$$

# Computation tree logic: Semantiek

Gegeven een Kripke structuur:

$$M = (S, S_0, R, L)$$

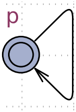
Formule $\phi$ is waar in state $s$ van de Kripke structuur $M$
*"Formula $\phi$ holds in state $s$ of Kripke structure $M$"*:

$$M, s \models \phi$$

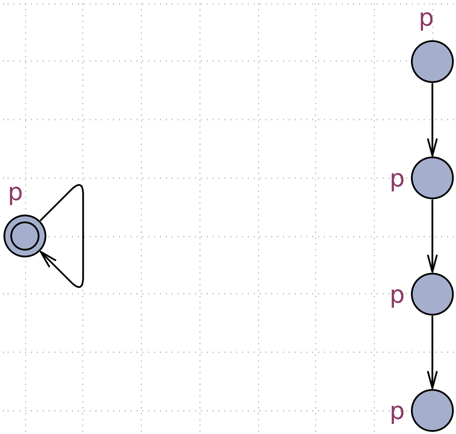
$p$ en $q$ zijn waar in state $s_1$ van Kripke structuur $M$:

$$M, s_1 \models p \wedge q$$

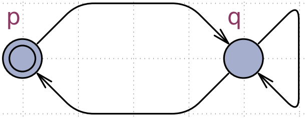
# De computation tree



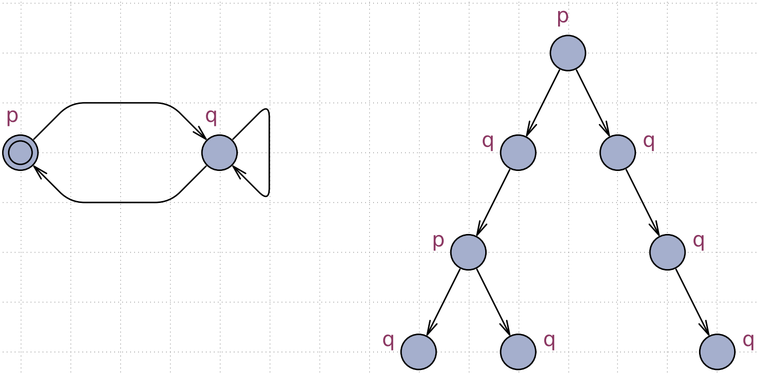
# De computation tree



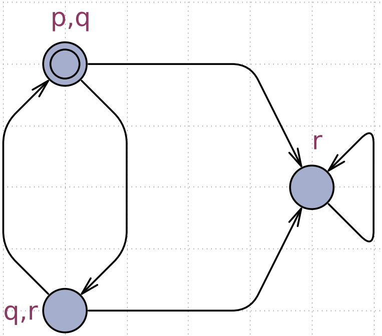
# De computation tree



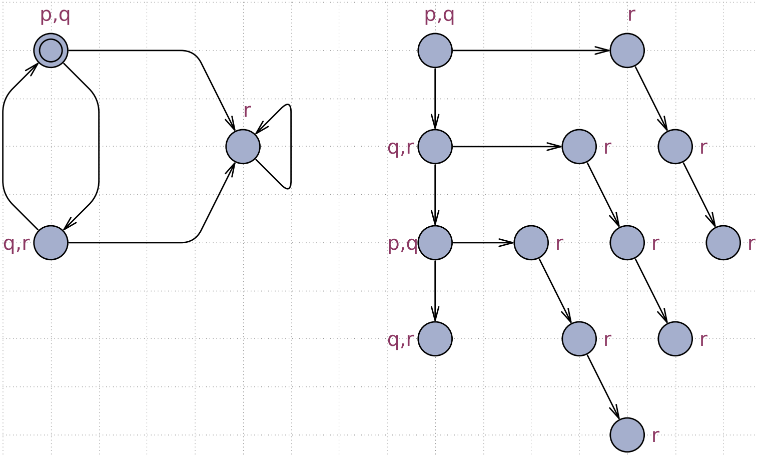
# De computation tree



# Oefening: Bouw de computation tree



# Oefening: Bouw de computation tree



# De computation tree

- De systemen die wij modelleren zijn reactief.

# De computation tree

- De systemen die wij modelleren zijn reactief.
- Derhalve is de transitierelatie in de Kripke structuur totaal.

# De computation tree

- De systemen die wij modelleren zijn reactief.
- Derhalve is de transitierelatie in de Kripke structuur totaal.
- Derhalve is de computation tree altijd oneindig groot.

# De computation tree

- De systemen die wij modelleren zijn reactief.
- Derhalve is de transitierelatie in de Kripke structuur totaal.
- Derhalve is de computation tree altijd oneindig groot.
- *"De computation tree heeft geen bladeren. . . "*

# Computation tree logic: semantiek

$\phi \equiv$

- $\phi | \neg\phi | \phi \wedge \phi | \phi \vee \phi |$
- $AG(\phi) | EG(\phi) |$
- $AF(\phi) | EF(\phi) |$
- $AX(\phi) | EX(\phi) |$
- $A(\phi U \phi) | E(\phi U \phi) |$
- $A(\phi R \phi) | E(\phi R \phi) |$

# Computation tree logic: Operatoren

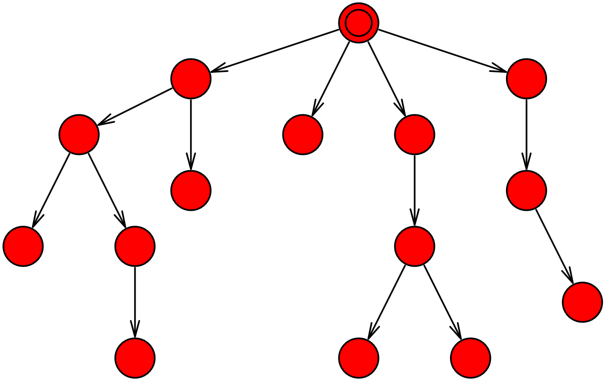
AG(p):

$$M, s \models AG(p) \iff \forall \pi \in \Pi(M, s) \cdot \forall i \cdot M, \pi[i] \models p$$

# Computation tree logic: Operatoren

AG(p):

$$M, s \models AG(p) \iff \forall \pi \in \Pi(M, s) \cdot \forall i \cdot M, \pi[i] \models p$$



# Computation tree logic: Operatoren

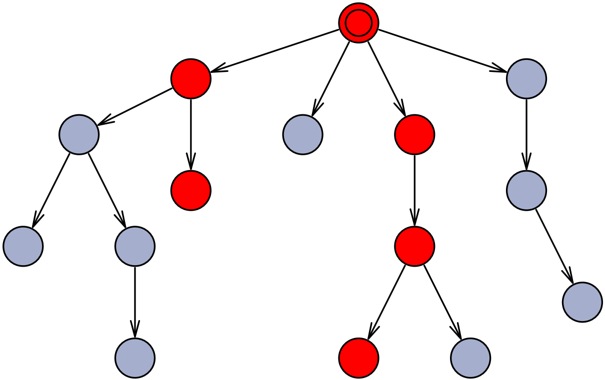
EG(p):

$$M, s \models EG(p) \iff \exists \pi \in \Pi(M, s) \cdot \forall i \cdot M, \pi[i] \models p$$

# Computation tree logic: Operatoren

EG(p):

$$M, s \models EG(p) \iff \exists \pi \in \Pi(M, s) \cdot \forall i \cdot M, \pi[i] \models p$$



# Computation tree logic: Operatoren

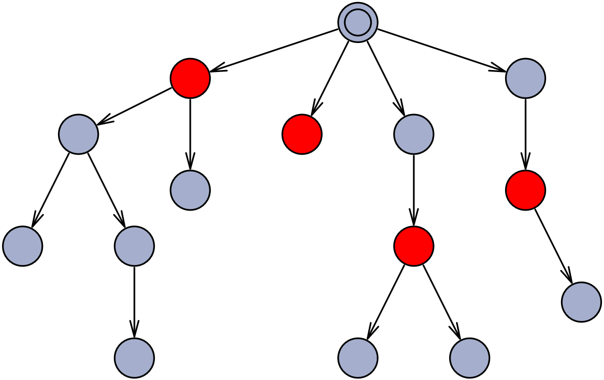
AF(p):

$$M, s \models AF(p) \iff \forall \pi \in \Pi(M, s) \cdot \exists i \cdot M, \pi[i] \models p$$

# Computation tree logic: Operatoren

AF(p):

$$M, s \models AF(p) \iff \forall \pi \in \Pi(M, s) \cdot \exists i \cdot M, \pi[i] \models p$$



# Computation tree logic: Operatoren

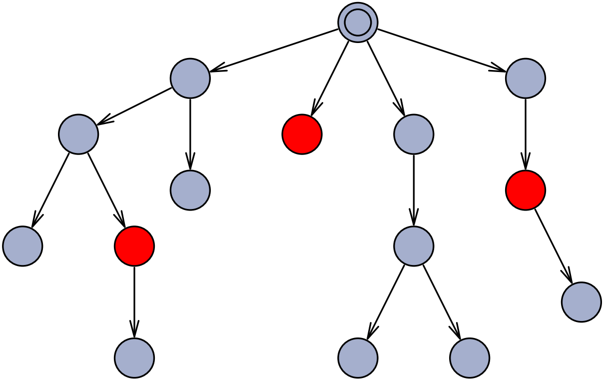
EF(p):

$$M, s \models EF(p) \iff \exists \pi \in \Pi(M, s) \cdot \exists i \cdot M, \pi[i] \models p$$

# Computation tree logic: Operatoren

EF(p):

$$M, s \models EF(p) \iff \exists \pi \in \Pi(M, s) \cdot \exists i \cdot M, \pi[i] \models p$$



# Computation tree logic: Operatoren

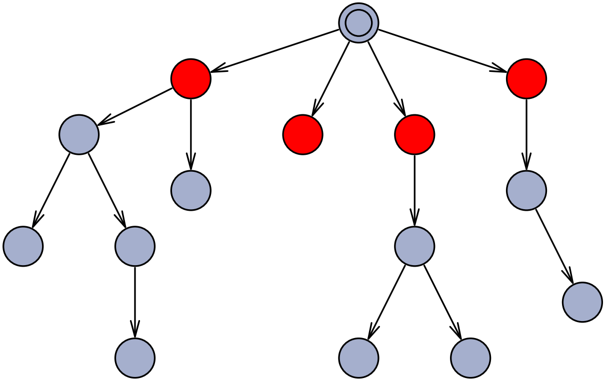
AX(p):

$$M, s \models AX(p) \iff \forall \pi \in \Pi(M, s) \cdot M, \pi[1] \models p$$

# Computation tree logic: Operatoren

AX(p):

$$M, s \models AX(p) \iff \forall \pi \in \Pi(M, s) \cdot M, \pi[1] \models p$$



# Computation tree logic: Operatoren

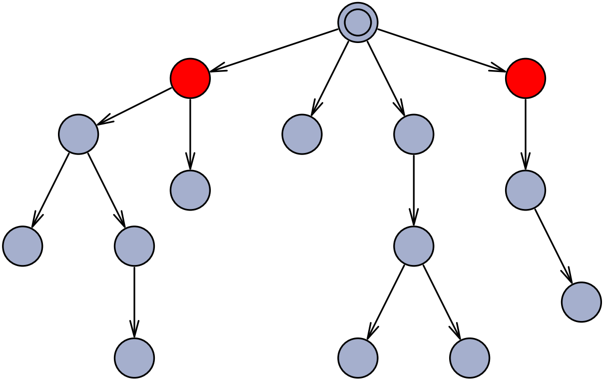
EX(p):

$$M, s \models EX(p) \iff \exists \pi \in \Pi(M, s) \cdot M, \pi[1] \models p$$

# Computation tree logic: Operatoren

EX(p):

$$M, s \models EX(p) \iff \exists \pi \in \Pi(M, s) \cdot M, \pi[1] \models p$$



# Computation tree logic: Operatoren

A(p U q):

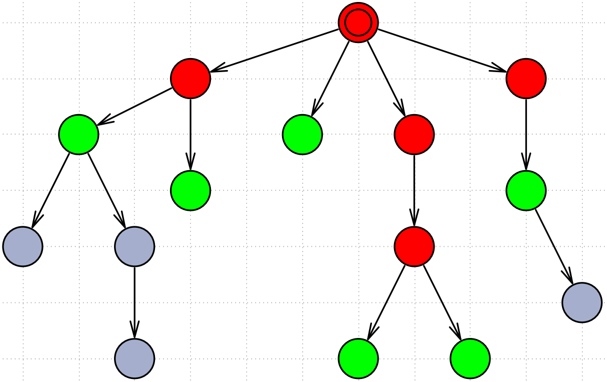
$$M, s \models A(pUq)$$

$$\iff$$

$$\forall \pi \in \Pi(M, s) \cdot \exists k \cdot (M, \pi[k] \models q) \wedge (\forall i < k \cdot M, \pi[i] \models p)$$

# Computation tree logic: Operatoren

A(p U q):



# Computation tree logic: Operatoren

E(p U q):

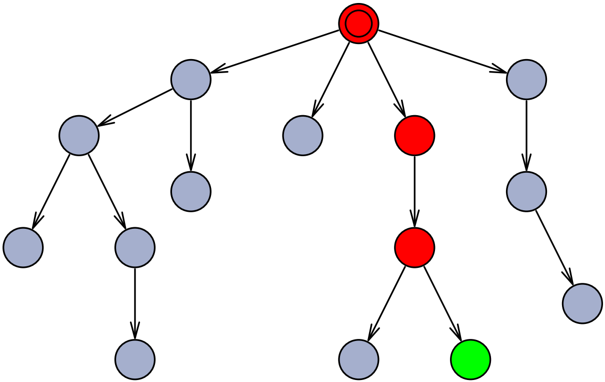
$$M, s \models E(pUq)$$

$$\iff$$

$$\exists \pi \in \Pi(M, s) \cdot \exists k \cdot (M, \pi[k] \models q) \wedge (\forall i < k \cdot M, \pi[i] \models p)$$

# Computation tree logic: Operatoren

E(p U q):



# Computation tree logic: Operatoren

A(p R q):

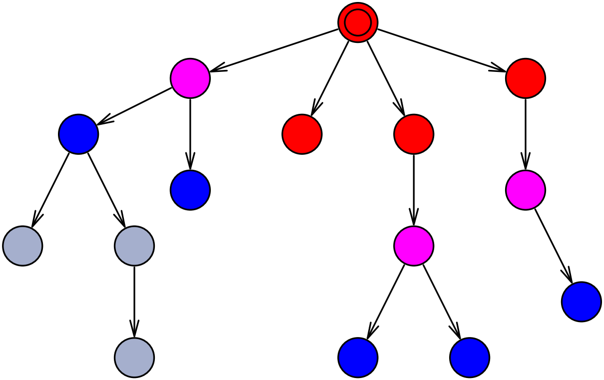
$$M, s \models A(pRq)$$

$$\iff$$

$$\forall \pi \in \Pi(M, s) \cdot \forall k \cdot (\forall i < k \cdot M, \pi[i] \models \neg p) = (M, \pi[k] \models q)$$

# Computation tree logic: Operatoren

A(p R q):



# Computation tree logic: Operatoren

E(p R q):

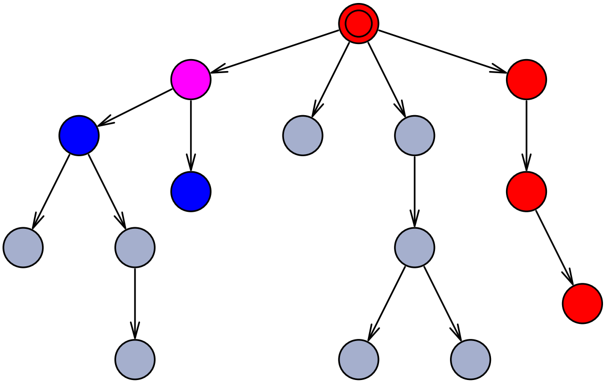
$$M, s \models E(pRq)$$

$$\iff$$

$$\exists \pi \in \Pi(M, s) \cdot \forall k \cdot (\forall i < k \cdot M, \pi[i] \models \neg p) = (M, \pi[k] \models q)$$

# Computation tree logic: Operatoren

E(p R q):



# Computation tree logic: Geneste operatoren

Nogmaals de semantiek:

$$\phi \equiv \phi | \neg\phi | \phi \wedge \phi | \phi \vee \phi | AG(\phi) | EG(\phi) | AF(\phi) | EF(\phi) | AX(\phi) | EX(\phi) |$$
$$A(\phi U \phi) | E(\phi U \phi) | A(\phi R \phi) | E(\phi R \phi) |$$

- Bovenstaande definitie is recursief. . .

# Computation tree logic: Geneste operatoren

Nogmaals de semantiek:

$$\begin{aligned}\phi \equiv &\phi|\neg\phi|\phi \wedge \phi|\phi \vee \phi|AG(\phi)|EG(\phi)|AF(\phi)|EF(\phi)|AX(\phi)|EX(\phi)| \\ &A(\phi U\phi)|E(\phi U\phi)|A(\phi R\phi)|E(\phi R\phi)|\end{aligned}$$

- Bovenstaande definitie is recursief...
- Derhalve ook mogelijk:

# Computation tree logic: Geneste operatoren

Nogmaals de semantiek:

$$\phi \equiv \phi|\neg\phi|\phi \wedge \phi|\phi \vee \phi|AG(\phi)|EG(\phi)|AF(\phi)|EF(\phi)|AX(\phi)|EX(\phi)|$$
$$A(\phi U \phi)|E(\phi U \phi)|A(\phi R \phi)|E(\phi R \phi)|$$

- Bovenstaande definitie is recursief...
- Derhalve ook mogelijk:
- $AG(AF(p))$
- $EF(AF(EX(p)))$
- $AG(EX(p))$
- ...

# Computation tree logic: Geneste operatoren

Combinaties van ctl operatoren zijn niet altijd zinvol, maar er zitten een aantal nuttige tussen:

- $AG(EF(p))$
- $AG(AF(p))$
- $AF(AG(p))$
- $EF(AG(p))$
- $AG(p \rightarrow AF(q))$

# Geneste operatoren: bereikbaarheid

$AG(EF(p))$:

# Geneste operatoren: bereikbaarheid

$AG(EF(p))$:

- Op elk denkbaar pad moet in elke denkbare state gelden $EF(p)$

# Geneste operatoren: bereikbaarheid

$AG(EF(p))$:

- Op elk denkbaar pad moet in elke denkbare state gelden $EF(p)$
- M.a.w. In welke state de automaat zich ook bevindt, er is altijd een pad te vinden naar een state, waarin $p$ geldig is.

# Geneste operatoren: bereikbaarheid

$AG(EF(p))$:

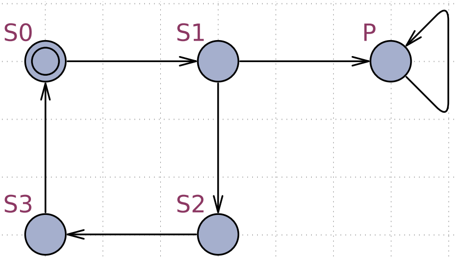
- Op elk denkbaar pad moet in elke denkbare state gelden $EF(p)$
- M.a.w. In welke state de automaat zich ook bevindt, er is altijd een pad te vinden naar een state, waarin $p$ geldig is.

Conclusie:

- $AG(EF(p))$ vertegenwoordigt een *bereikbaarheidsuitspraak*: Een toestand, waarin $p$ geldt is altijd bereikbaar.

# Geneste operatoren: bereikbaarheid

$AG(EF(p))$:



# Fairness

$AG(AF(p))$:

# Fairness

$AG(AF(p))$:

- Op elk denkbaar pad moet in elke denkbare state gelden $AF(p)$

# Fairness

$AG(AF(p))$:

- Op elk denkbaar pad moet in elke denkbare state gelden $AF(p)$
- M.a.w. In welke state de automaat zich ook bevindt, in alle richtingen kom je vroeg of laat een state tegen, waarin $p$ geldig is.

# Fairness

$AG(AF(p))$:

- Op elk denkbaar pad moet in elke denkbare state gelden $AF(p)$
- M.a.w. In welke state de automaat zich ook bevindt, in alle richtingen kom je vroeg of laat een state tegen, waarin $p$ geldig is.
- M.a.w. *"p vindt oneindig vaak plaats."*

# Fairness

$AG(AF(p))$:

- Op elk denkbaar pad moet in elke denkbare state gelden $AF(p)$
- M.a.w. In welke state de automaat zich ook bevindt, in alle richtingen kom je vroeg of laat een state tegen, waarin $p$ geldig is.
- M.a.w. *"p vindt oneindig vaak plaats."*

$AG(AF(p))$ staat bekend als *fairness* of *freedom of starvation*.

## Geneste operatoren

$AF(AG(p))$:

# Geneste operatoren

$AF(AG(p))$:

- Vroeg of laat zal $p$ gelden en blijven gelden.

# Geneste operatoren

$AF(AG(p))$:

- Vroeg of laat zal $p$ gelden en blijven gelden.
- Voorbeeld: *"Na het opstarten blijft het groene controlelampje branden..."*

# Liveness (1/3)

$AG(p \to AF(q))$:

# Liveness (1/3)

$AG(p \rightarrow AF(q))$:

- Altijd en overal geldt: $p \rightarrow AF(q)$

# Liveness (1/3)

$AG(p \rightarrow AF(q))$:

- Altijd en overal geldt: $p \rightarrow AF(q)$
- Altijd en overal geldt: *"Als p geldt dan geldt vroeg of laat q"*

# Liveness (1/3)

$AG(p \rightarrow AF(q))$:

- Altijd en overal geldt: $p \rightarrow AF(q)$
- Altijd en overal geldt: *"Als p geldt dan geldt vroeg of laat q"*
- De $AG(p \rightarrow AF(q))$ eigenschap noemt men *liveness*.

# Liveness (2/3)

$AG(p \rightarrow AF(q))$
Opmerking 1:

# Liveness (2/3)

$AG(p \rightarrow AF(q))$

Opmerking 1:

- Vul voor $p$ simpelweg `true` in.

# Liveness (2/3)

$AG(p \rightarrow AF(q))$
Opmerking 1:

- Vul voor $p$ simpelweg `true` in.
- We krijgen nu: $AG(true \rightarrow AF(q))$

# Liveness (2/3)

$AG(p \to AF(q))$
Opmerking 1:

- Vul voor $p$ simpelweg `true` in.
- We krijgen nu: $AG(true \to AF(q))$
- En dus: $AG(AF(q))$

# Liveness (2/3)

$AG(p \rightarrow AF(q))$

Opmerking 1:

- Vul voor $p$ simpelweg `true` in.
- We krijgen nu: $AG(true \rightarrow AF(q))$
- En dus: $AG(AF(q))$
- Conclusie: liveness is een generalisatie van fairness...

# Liveness (2/3)

$AG(p \rightarrow AF(q))$
Opmerking 1:

- Vul voor $p$ simpelweg `true` in.
- We krijgen nu: $AG(true \rightarrow AF(q))$
- En dus: $AG(AF(q))$
- Conclusie: liveness is een generalisatie van fairness...

Opmerking 2:

# Liveness (2/3)

$AG(p \rightarrow AF(q))$

Opmerking 1:

- Vul voor $p$ simpelweg `true` in.
- We krijgen nu: $AG(true \rightarrow AF(q))$
- En dus: $AG(AF(q))$
- Conclusie: liveness is een generalisatie van fairness…

Opmerking 2:

- Liveness heeft een speciale operator:
- $p \rightsquigarrow q \equiv AG(p \rightarrow AF(q))$

# Liveness (3/3)

$p \rightsquigarrow q \equiv AG(p \rightarrow AF(q))$

- Gegeven een systeem, waarin $p$ nooit optreedt...

# Liveness (3/3)

$p \leadsto q \equiv AG(p \rightarrow AF(q))$

- Gegeven een systeem, waarin $p$ nooit optreedt...
- Geldt nu de liveness eigenschap?

# Liveness (3/3)

$p \rightsquigarrow q \equiv AG(p \rightarrow AF(q))$

- Gegeven een systeem, waarin $p$ nooit optreedt...
- Geldt nu de liveness eigenschap?
- Antwoord: Ja

# Liveness (3/3)

$$p \leadsto q \equiv AG(p \rightarrow AF(q))$$

- Gegeven een systeem, waarin $p$ nooit optreedt...
- Geldt nu de liveness eigenschap?
- Antwoord: Ja, immers: $\models 1 \Leftrightarrow 0 \rightarrow 0$

# Liveness (3/3)

$p \leadsto q \equiv AG(p \rightarrow AF(q))$

- Gegeven een systeem, waarin $p$ nooit optreedt. . .
- Geldt nu de liveness eigenschap?
- Antwoord: Ja, immers: $\models 1 \Leftrightarrow 0 \rightarrow 0$

In een situatie, waarin $p$ nooit optreedt, spreekt men van een *vacuous truth*.

# Liveness (3/3)

$p \rightsquigarrow q \equiv AG(p \rightarrow AF(q))$

- Gegeven een systeem, waarin $p$ nooit optreedt. . .
- Geldt nu de liveness eigenschap?
- Antwoord: Ja, immers: $\models 1 \Leftrightarrow 0 \rightarrow 0$

In een situatie, waarin $p$ nooit optreedt, spreekt men van een *vacuous truth*.
Voorbeeld:

- *"Als Hans Teeuwen minister president wordt, eet ik mijn schoenen op."*

# Liveness (3/3)

$p \rightsquigarrow q \equiv AG(p \rightarrow AF(q))$

- Gegeven een systeem, waarin $p$ nooit optreedt...
- Geldt nu de liveness eigenschap?
- Antwoord: Ja, immers: $\models 1 \Leftrightarrow 0 \rightarrow 0$

In een situatie, waarin $p$ nooit optreedt, spreekt men van een *vacuous truth*.
Voorbeeld:

- *"Als Hans Teeuwen minister president wordt, eet ik mijn schoenen op."*
- *"Als de VS een nucleaire aanval op Rusland uitvoeren, slaat Rusland nucleair terug... "*

## Liveness (3/3)

$p \rightsquigarrow q \equiv AG(p \rightarrow AF(q))$

- Gegeven een systeem, waarin $p$ nooit optreedt...
- Geldt nu de liveness eigenschap?
- Antwoord: Ja, immers: $\models 1 \Leftrightarrow 0 \rightarrow 0$

In een situatie, waarin $p$ nooit optreedt, spreekt men van een *vacuous truth*.
Voorbeeld:

- *"Als Hans Teeuwen minister president wordt, eet ik mijn schoenen op."*
- *"Als de VS een nucleaire aanval op Rusland uitvoeren, slaat Rusland nucleair terug... "*
- Zoek op: de M.A.D. doctrine $\rightarrow$ Nash equilibrium