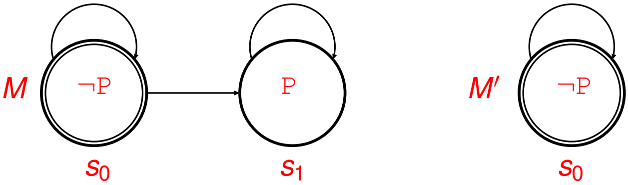
# A property not expressible in LTL

- Let $AP = \{\mathtt{P}\}$ and consider models $M$ and $M'$ below




$$\begin{array}{lcl} M & = & (\{s_0, s_1\}, \{s_0\}, \{(s_0, s_0), (s_0, s_1), (s_1, s_1)\}, L) \\ M' & = & (\{s_0\}, \{s_0\}, \{(s_0, s_0)\}, L) \end{array}$$

where: $L \;=\; \lambda s.\ \mathit{if}\ s = s_0\ \mathit{then}\ \{\}\ \mathit{else}\ \{\mathtt{P}\}$

- Every $M'$-path is also an $M$-path
- So if $\phi$ true on every $M$-path then $\phi$ true on every $M'$-path
- Hence in LTL for any $\phi$ if $M \models \phi$ then $M' \models \phi$
- Consider $\phi_{\mathtt{P}} \Leftrightarrow$ "can always reach a state satisfying $\mathtt{P}$"
  - $\phi_{\mathtt{P}}$ holds in $M$ but not in $M'$
  - but in LTL can't have $M \models \phi_{\mathtt{P}}$ and not $M' \models \phi_{\mathtt{P}}$
- hence $\phi_{\mathtt{P}}$ not expressible in LTL

# CTL model checking

- For LTL path formulae $\phi$ recall that $M \models \phi$ is defined by:

  $$\boxed{M \models \phi \;\Leftrightarrow\; \forall \pi\ s.\ s \in S_0 \wedge \mathrm{Path}\ R\ s\ \pi \Rightarrow [\![\phi]\!]_M(\pi)}$$

- For CTL state formulae $\psi$ the definition of $M \models \psi$ is:

  $$\boxed{M \models \psi \;\Leftrightarrow\; \forall s.\ s \in S_0 \Rightarrow [\![\psi]\!]_M(s)}$$

- $M$ common; LTL, CTL formulae and semantics $[\![\ ]\!]_M$ differ

- CTL model checking algorithm:
  - compute $\{s \mid [\![\psi]\!]_M(s) = \mathit{true}\}$ bottom up
  - check $S_0 \subseteq \{s \mid [\![\psi]\!]_M(s) = \mathit{true}\}$
  - symbolic model checking represents these sets as BDDs

# CTL model checking: $p$, $\mathbf{AX}\psi$, $\mathbf{EX}\psi$

- For CTL formula $\psi$ let $\{\!|\psi|\!\}_M = \{s \mid [\![\psi]\!]_M(s) = \mathit{true}\}$
- When unambiguous will write $\{\!|\psi|\!\}$ instead of $\{\!|\psi|\!\}_M$
- $\{\!|p|\!\} = \{s \mid p \in L(s)\}$
  - scan through set of states $S$ marking states labelled with $p$
  - $\{\!|p|\!\}$ is set of marked states
- To compute $\{\!|\mathbf{AX}\psi|\!\}$
  - recursively compute $\{\!|\psi|\!\}$
  - marks those states all of whose successors are in $\{\!|\psi|\!\}$
  - $\{\!|\mathbf{AX}\psi|\!\}$ is the set of marked states
- To compute $\{\!|\mathbf{EX}\psi|\!\}$
  - recursively compute $\{\!|\psi|\!\}$
  - marks those states with at least one successor in $\{\!|\psi|\!\}$
  - $\{\!|\mathbf{EX}\psi|\!\}$ is the set of marked states

# CTL model checking: $[\![\mathbf{E}[\psi_1\ \mathbf{U}\ \psi_2]]\!]$, $[\![\mathbf{A}[\psi_1\ \mathbf{U}\ \psi_2]]\!]$

- To compute $[\![\mathbf{E}[\psi_1\ \mathbf{U}\ \psi_2]]\!]$
  - recursively compute $[\![\psi_1]\!]$ and $[\![\psi_2]\!]$
  - mark all states in $[\![\psi_2]\!]$
  - mark all states in $[\![\psi_1]\!]$ with a successor state that is marked
  - repeat previous line until no change
  - $[\![\mathbf{E}[\psi_1\ \mathbf{U}\ \psi_2]]\!]$ is set of marked states

- More formally: $[\![\mathbf{E}[\psi_1\ \mathbf{U}\ \psi_2]]\!] = \bigcup_{n=0}^{\infty} [\![\mathbf{E}[\psi_1\ \mathbf{U}\ \psi_2]]\!]_n$ where:

$$
\begin{array}{rcl}
[\![\mathbf{E}[\psi_1\ \mathbf{U}\ \psi_2]]\!]_0 & = & [\![\psi_2]\!] \\
[\![\mathbf{E}[\psi_1\ \mathbf{U}\ \psi_2]]\!]_{n+1} & = & [\![\mathbf{E}[\psi_1\ \mathbf{U}\ \psi_2]]\!]_n \\
 & & \cup \\
 & & \{s \in [\![\psi_1]\!] \mid \exists s' \in [\![\mathbf{E}[\psi_1\ \mathbf{U}\ \psi_2]]\!]_n.\ R\ s\ s'\}
\end{array}
$$

- $[\![\mathbf{A}[\psi_1\ \mathbf{U}\ \psi_2]]\!]$ similar, but with a more complicated iteration
  - details omitted (see Huth and Ryan)

# Example: checking EF $p$

- $\mathbf{EF}p = \mathbf{E}[\mathtt{T}\ \mathbf{U}\ p]$
  - holds if $\psi$ holds along some path

- Note $[\![\mathtt{T}]\!] = S$

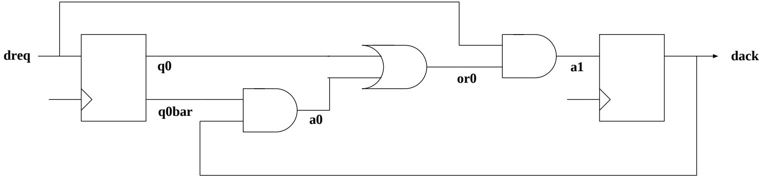
- Let $\mathcal{S}_n = [\![\mathbf{E}[\mathtt{T}\ \mathbf{U}\ p]]\!]_n$ then:

$$\begin{array}{lcl}
\mathcal{S}_0 & = & [\![\mathbf{E}[\mathtt{T}\ \mathbf{U}\ p]]\!]_0 \\
 & = & [\![p]\!] \\
 & = & \{s \mid p \in L(s)\} \\
\mathcal{S}_{n+1} & = & \mathcal{S}_n\ \cup\ \{s \in [\![\mathtt{T}]\!] \mid \exists s' \in [\![\mathbf{E}[\mathtt{T}\ \mathbf{U}\ p]]\!]_n.\ R\ s\ s'\} \\
 & = & \mathcal{S}_n\ \cup\ \{s \mid \exists s' \in \mathcal{S}_n.\ R\ s\ s'\}
\end{array}$$

  - mark all the states labelled with $p$
  - mark all with at least one marked successor
  - repeat until no change
  - $[\![\mathbf{EF}\ p]\!]$ is set of marked states

# Example: RCV

- Recall the handshake circuit:




- State represented by a triple of Booleans $(dreq, q0, dack)$
- A model of RCV is $M_{\mathrm{RCV}}$ where:

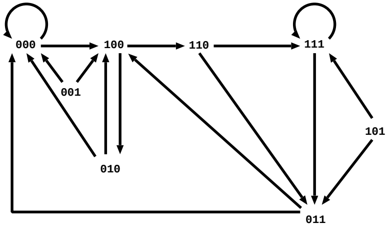
  $M = (S_{\mathrm{RCV}}, S0_{\mathrm{RCV}}, R_{\mathrm{RCV}}, L_{\mathrm{RCV}})$

  and

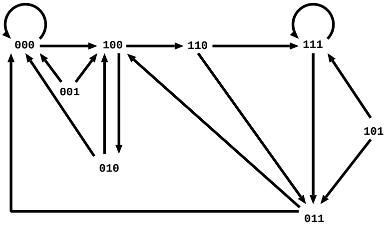
  $$R_{\mathrm{RCV}}\ (dreq, q0, dack)\ (dreq', q0', dack') = \\ (q0' = dreq) \land (dack' = (dreq \land (q0 \lor dack)))$$

# RCV state transition diagram

- Possible states for RCV:
  $\{000, 001, 010, 011, 100, 101, 110, 111\}$
  where $b_2 b_1 b_0$ denotes state
  $\mathtt{dreq} = b_2 \ \wedge\ \mathtt{q0} = b_1 \ \wedge\ \mathtt{dack} = b_0$
- Graph of the transition relation:



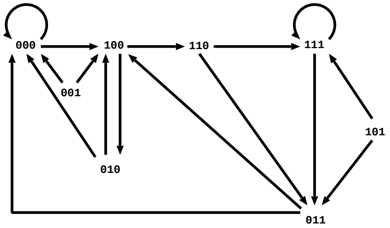
# Computing Reachable $M_{\mathrm{RCV}}$




- Define:

$$\begin{aligned}
\mathcal{S}_0 &= \{b_2 b_1 b_0 \mid b_2 b_1 b_0 \in \{111\}\} \\
&= \{111\} \\
\mathcal{S}_{i+1} &= \mathcal{S}_i \ \cup \ \{s' \mid \exists s \in \mathcal{S}_i .\ R_{\mathrm{RCV}}\ s\ s'\ \} \\
&= \mathcal{S}_i \ \cup \ \{b_2' b_1' b_0' \mid \\
&\qquad \exists b_2 b_1 b_0 \in \mathcal{S}_i .\ (b_1' = b_2) \ \wedge \ (b_0' = b_2 \wedge (b_1 \vee b_0))\}
\end{aligned}$$

# Computing $\{\mathbf{EF}\ \mathtt{At111}\}$ where $\mathtt{At111} \in L_{\mathrm{RCV}}(s) \Leftrightarrow s = 111$

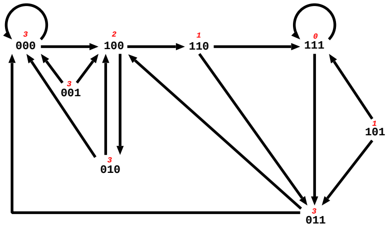



- Define:

$$
\begin{aligned}
\mathcal{S}_0 &= \{s \mid \mathtt{At111} \in L_{\mathrm{RCV}}(s)\} \\
&= \{s \mid s = 111\} \\
&= \{111\}
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{S}_{n+1} &= \mathcal{S}_n \ \cup \ \{s \mid \exists s' \in \mathcal{S}_n.\ \mathcal{R}(s, s')\} \\
&= \mathcal{S}_n \ \cup \ \{b_2 b_1 b_0 \mid \\
&\qquad \exists b_2' b_1' b_0' \in \mathcal{S}_n.\ (b_1' = b_2) \ \wedge \ (b_0' = b_2 \wedge (b_1 \vee b_0))\}
\end{aligned}
$$

# Computing {**EF** `At111`} (continued)




- Compute:

$$
\begin{aligned}
\mathcal{S}_0 &= \{111\} \\
\mathcal{S}_1 &= \{111\} \cup \{101, 110\} \\
&= \{111, 101, 110\} \\
\mathcal{S}_2 &= \{111, 101, 110\} \cup \{100\} \\
&= \{111, 101, 110, 100\} \\
\mathcal{S}_3 &= \{111, 101, 110, 100\} \cup \{000, 001, 010, 011\} \\
&= \{111, 101, 110, 100, 000, 001, 010, 011\} \\
\mathcal{S}_n &= \mathcal{S}_3 \quad (n > 3)
\end{aligned}
$$

- $\{\mathbf{EF}\ \mathtt{At111}\} = \mathbb{B}^3 = S_{\mathtt{RCV}}$
- $M_{\mathtt{RCV}} \models \mathbf{EF}\ \mathtt{At111} \Leftrightarrow S_{0\,\mathtt{RCV}} \subseteq S$

# Symbolic model checking

- Represent sets of states with BDDs
- Represent Transition relation with a BDD
- If BDDs of $\{\psi\}$, $\{\psi_1\}$, $\{\psi_2\}$ are known, then:
  - BDDs of $\{\neg\psi\}$, $\{\psi_1 \wedge \psi_2\}$, $\{\psi_1 \vee \psi_2\}$, $\{\psi_1 \Rightarrow \psi_2\}$ computed using standard BDD algorithms
  - BDDs of $\{\mathbf{AX}\psi\}$, $\{\mathbf{EX}\psi\}$, $\{\mathbf{A}[\psi_1\ \mathbf{U}\ \psi_2]\}$, $\{\mathbf{E}[\psi_1\ \mathbf{U}\ \psi_2]]\}$ computed using straightforward algorithms (see textbooks)
- Model checking CTL generalises reachable states iteration

# History of Model checking

- CTL model checking due to Emerson, Clarke & Sifakis
- Symbolic model checking due to several people:
  - Clarke & McMillan (idea usually credited to McMillan's PhD)
  - Coudert, Berthet & Madre
  - Pixley
- SMV (McMillan) is a popular symbolic model checker:

| | |
|---|---|
| `http://www.cs.cmu.edu/~modelcheck/smv.html` | (original) |
| `http://www.kenmcmil.com/smv.html` | (Cadence extension by McMillan) |
| `http://nusmv.irst.itc.it/` | (new implementation) |

- Other temporal logics
  - CTL*: combines CTL and LTL
  - Engineer friendly industrial languages: PSL, SVA

# Expressibility of CTL

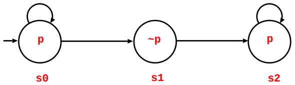
- Consider the property

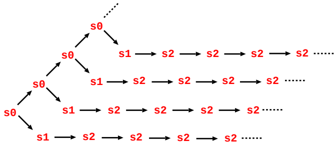
  *"on every path there is a point after which $p$ is always true on that path"*

- Consider

  (($\star$) non-deterministically chooses `T` or `F`)

```
      0:   P:=1;
  s0  1:   WHILE (*) DO SKIP;
  s1  2:   P:=0;
  s2  3:   P:=1;
      4:   WHILE T DO SKIP;
      5:
```




- Property true, but cannot be expressed in CTL
  - would need something like **AF**$\psi$
  - where $\psi$ is something like "property $\mathtt{p}$ *true from now on*"
  - but in CTL $\psi$ must start with a path quantifier **A** or **E**
  - cannot talk about current path, only about all or some paths
  - **AF(AG p)** is false (consider path $\mathtt{s0\,s0\,s0} \cdots$)

# LTL can express things CTL can't

- Recall:
  $$\begin{array}{lcl} [\![\mathbf{F}\phi]\!]_M(\pi) & = & \exists i.\ [\![\phi]\!]_M(\pi{\downarrow}i) \\ [\![\mathbf{G}\phi]\!]_M(\pi) & = & \forall i.\ [\![\phi]\!]_M(\pi{\downarrow}i) \end{array}$$

- $\mathbf{FG}\phi$ is true if there is a point after which $\phi$ is always true
  $$\begin{array}{ll} [\![\mathbf{FG}\phi]\!]_M(\pi) & = [\![\mathbf{F}(\mathbf{G}(\phi))]\!]_M(\pi) \\ & = \exists m_1.\ [\![\mathbf{G}(\phi)]\!]_M(\pi{\downarrow}m_1) \\ & = \exists m_1.\ \forall m_2.\ [\![\phi]\!]_M((\pi{\downarrow}m_1){\downarrow}m_2) \\ & = \exists m_1.\ \forall m_2.\ [\![\phi]\!]_M(\pi{\downarrow}(m_1+m_2)) \end{array}$$

- LTL can express things that CTL can't express

- Note: it's tricky to prove CTL can't express $\mathbf{FG}\phi$

# CTL can express things that LTL can't express

- **AG**(**EF** *p*) says:

  *"from every state it is possible to get to a state for which p holds"*

- Can't say this in LTL (easy proof given earlier - slide 57)

- Consider disjunction:

  *"on every path there is a point after which p is always true on that path*
  *or*
  *from every state it is possible to get to a state for which p holds"*

- Can't say this in either CTL or LTL!

- CTL* combines CTL and LTL and can express this property

# CTL*

- Both state formulae ($\psi$) and path formulae ($\phi$)
  - state formulae $\psi$ are true of a state $s$ like CTL
  - path formulae $\phi$ are true of a path $\pi$ like LTL
- Defined mutually recursively

$$
\begin{array}{lcll}
\psi & ::= & p & \text{(Atomic formula)} \\
 & | & \neg\psi & \text{(Negation)} \\
 & | & \psi_1 \vee \psi_2 & \text{(Disjunction)} \\
 & | & \mathbf{A}\phi & \text{(All paths)} \\
 & | & \mathbf{E}\phi & \text{(Some paths)} \\
\\
\phi & ::= & \psi & \text{(Every state formula is a path formula)} \\
 & | & \neg\phi & \text{(Negation)} \\
 & | & \phi_1 \vee \phi_2 & \text{(Disjunction)} \\
 & | & \mathbf{X}\phi & \text{(Successor)} \\
 & | & \mathbf{F}\phi & \text{(Sometimes)} \\
 & | & \mathbf{G}\phi & \text{(Always)} \\
 & | & [\phi_1\ \mathbf{U}\ \phi_2] & \text{(Until)}
\end{array}
$$

- CTL is CTL* with **X**, **F**, **G**, $[-\mathbf{U}-]$ preceded by **A** or **E**
- LTL consists of CTL* formulae of form $\mathbf{A}\phi$, where the only state formulae in $\phi$ are atomic

# CTL* semantics

- Combines CTL state semantics with LTL path semantics:

$$\begin{array}{lcl}
[\![p]\!]_M(s) & = & p \in L(s) \\
[\![\neg\psi]\!]_M(s) & = & \neg([\![\psi]\!]_M(s)) \\
[\![\psi_1 \vee \psi_2]\!]_M(s) & = & [\![\psi_1]\!]_M(s) \;\vee\; [\![\psi_2]\!]_M(s) \\
[\![\mathbf{A}\phi]\!]_M(s) & = & \forall\pi.\ \mathrm{Path}\ R\ s\ \pi \;\Rightarrow\; \phi(\pi) \\
[\![\mathbf{E}\phi]\!]_M(s) & = & \exists\pi.\ \mathrm{Path}\ R\ s\ \pi \;\wedge\; [\![\phi]\!]_M(\pi)
\end{array}$$

$$\begin{array}{lcl}
[\![\psi]\!]_M(\pi) & = & [\![\psi]\!]_M(\pi(0)) \\
[\![\neg\phi]\!]_M(\pi) & = & \neg([\![\phi]\!]_M(\pi)) \\
[\![\phi_1 \vee \phi_2]\!]_M(\pi) & = & [\![\phi_1]\!]_M(\pi) \;\vee\; [\![\phi_2]\!]_M(\pi) \\
[\![\mathbf{X}\phi]\!]_M(\pi) & = & [\![\phi]\!]_M(\pi{\downarrow}1) \\
[\![\mathbf{F}\phi]\!]_M(\pi) & = & \exists m.\ [\![\phi]\!]_M(\pi{\downarrow}m) \\
[\![\mathbf{G}\phi]\!]_M(\pi) & = & \forall m.\ [\![\phi]\!]_M(\pi{\downarrow}m) \\
[\![[\phi_1\ \mathbf{U}\ \phi_2]]\!]_M(\pi) & = & \exists i.\ [\![\phi_2]\!]_M(\pi{\downarrow}i) \wedge \forall j.\ j{<}i \;\Rightarrow\; [\![\phi_1]\!]_M(\pi{\downarrow}j)
\end{array}$$

- Note $[\![\psi]\!]_M : S{\to}\mathbb{B}$ and $[\![\phi]\!]_M : (\mathbb{N}{\to}S){\to}\mathbb{B}$

# LTL and CTL as CTL*

- As usual: $M = (S, S_0, R, L)$
- If $\psi$ is a CTL* state formula: $M \models \psi \;\Leftrightarrow\; \forall s \in S_0.\ [\![\psi]\!]_M(s)$
- If $\phi$ is an LTL path formula then: $M \models_{\mathrm{LTL}} \phi \;\Leftrightarrow\; M \models_{\mathrm{CTL^*}} \mathbf{A}\phi$
- If $R$ is total ($\forall s.\ \exists s'.\ R\ s\ s'$) then (exercise):
  $\forall s\ s'.\ R\ s\ s' \;\Leftrightarrow\; \exists \pi.\ \mathrm{Path}\ R\ s\ \pi \wedge (\pi(1) = s')$
- The meanings of CTL formulae are the same in CTL*

$$
\begin{array}{ll}
[\![\mathbf{A}(\mathbf{X}\psi)]\!]_M(s) & \\
= \forall \pi.\ \mathrm{Path}\ R\ s\ \pi \Rightarrow [\![\mathbf{X}\psi]\!]_M(\pi) & \\
= \forall \pi.\ \mathrm{Path}\ R\ s\ \pi \Rightarrow [\![\psi]\!]_M(\pi{\downarrow}1) & (\psi \text{ as path formula}) \\
= \forall \pi.\ \mathrm{Path}\ R\ s\ \pi \Rightarrow [\![\psi]\!]_M((\pi{\downarrow}1)(0)) & (\psi \text{ as state formula}) \\
= \forall \pi.\ \mathrm{Path}\ R\ s\ \pi \Rightarrow [\![\psi]\!]_M(\pi(1)) &
\end{array}
$$

$$
\begin{array}{l}
[\![\mathbf{AX}\psi]\!]_M(s) \\
= \forall s'.\ R\ s\ s' \;\Rightarrow\; [\![\psi]\!]_M(s') \\
= \forall s'.\ (\exists \pi.\ \mathrm{Path}\ R\ s\ \pi \wedge (\pi(1) = s')) \;\Rightarrow\; [\![\psi]\!]_M(s') \\
= \forall s'.\ \forall \pi.\ \mathrm{Path}\ R\ s\ \pi \wedge (\pi(1) = s') \;\Rightarrow\; [\![\psi]\!]_M(s') \\
= \forall \pi.\ \mathrm{Path}\ R\ s\ \pi \;\Rightarrow\; [\![\psi]\!]_M(\pi(1))
\end{array}
$$

Exercise: do similar proofs for other CTL formulae

# Fairness

- May want to assume system or environment is ‘fair’
- Example 1: fair arbiter
  the arbiter doesn’t ignore one of its requests forever
  - not every request need be granted
  - want to exclude infinite number of requests and no grant
- Example 2: reliable channel
  no message continuously transmitted but never received
  - not every message need be received
  - want to exclude an infinite number of sends and no receive

# Handling fairness in CTL and LTL

- Consider:
  *p holds infinitely often along a path then so does q*
- In LTL is expressible as $\mathbf{G}(\mathbf{F}\ p) \Rightarrow \mathbf{G}(\mathbf{F}\ q)$
- Can't say this in CTL
  - why not – what's wrong with $\mathbf{AG}(\mathbf{AF}\ p) \Rightarrow \mathbf{AG}(\mathbf{AF}\ q)$?
  - in CTL* expressible as $\mathbf{A}(\mathbf{G}(\mathbf{F}\ p) \Rightarrow \mathbf{G}(\mathbf{F}\ q))$
  - fair CTL model checking implemented in checking algorithm
  - fair LTL just a fairness assumption like $\mathbf{G}(\mathbf{F}\ p) \Rightarrow \cdots$
- Fairness is a tricky and subtle subject
  - many kinds of fairness:
    'weak fairness', 'strong fairness' etc
  - exist whole books on fairness

# Propositional modal $\mu$-calculus

- You may learn this in *Topics in Concurrency*
- $\mu$-calculus is an even more powerful property language
  - has fixed-point operators
  - both maximal and minimal fixed points
  - model checking consists of calculating fixed points
  - many logics (e.g. CTL*) can be translated into $\mu$-calculus
- Strictly stronger than CTL*
  - expressibility strictly increases as allowed nesting increases
  - need fixed point operators nested 2 deep for CTL*
- The $\mu$-calculus is very non-intuitive to use!
  - intermediate code rather than a practical property language
  - nice meta-theory and algorithms, but terrible usability!