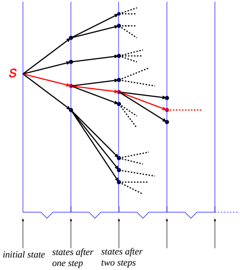
Model behaviour viewed as a computation tree

- Atomic properties are true or false of individual states
- General properties are true or false of whole behaviour
- Behaviour of (S, R) starting from $s \in S$ as a tree:



- A path is shown in red
- Properties may look at all paths, or just a single path
  - CTL: Computation Tree Logic (all paths from a state)
  - LTL: Linear Temporal Logic (a single path)

Paths

- A path of (S, R) is represented by a function $\pi : \mathbb{N} \to S$
  - $\pi(i)$ is the ith element of π (first element is $\pi(0)$)
  - might sometimes write $\pi\ i$ instead of $\pi(i)$
  - $\pi{\downarrow}i$ is the i-th tail of π so $\pi{\downarrow}i(n) = \pi(i+n)$
  - successive states in a path must be related by R

- Path $R\ s\ \pi$ is true if and only if π is a path starting at s:

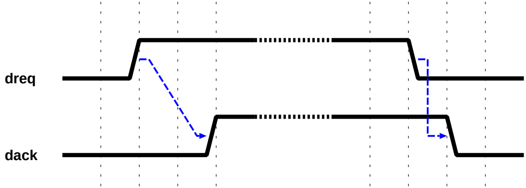
  $$\text{Path}\ R\ s\ \pi\ =\ (\pi(0) = s)\ \wedge\ \forall i.\ R\ (\pi(i))\ (\pi(i{+}1))$$

  where:

  $$\text{Path} : \underbrace{(S \to S \to \mathbb{B})}_{\text{transition relation}} \to \underbrace{S}_{\text{initial state}} \to \underbrace{(\mathbb{N} \to S)}_{\text{path}} \to \mathbb{B}$$

RCV: example hardware properties

- Consider this timing diagram:



- Two handshake properties representing the diagram:
  - following a rising edge on `dreq`, the value of `dreq` remains 1 (i.e. *true*) until it is acknowledged by a rising edge on `dack`
  - following a falling edge on `dreq`, the value on `dreq` remains 0 (i.e. *false*) until the value of `dack` is 0
- A property language is used to formalise such properties

DIV: example program properties

```
0:   R:=X;
1:   Q:=0;
2:   WHILE Y≤R DO
3:   (R:=R-Y;
4:    Q:=Q+1)
5:
```

$$
\begin{array}{lcl}
\texttt{AtStart}\,(pc, x, y, r, q) & = & (pc = 0) \\
\texttt{AtEnd}\,(pc, x, y, r, q) & = & (pc = 5) \\
\texttt{InLoop}\,(pc, x, y, r, q) & = & (pc \in \{3, 4\}) \\
\texttt{YleqR}\,(pc, x, y, r, q) & = & (y \leq r) \\
\texttt{Invariant}\,(pc, x, y, r, q) & = & (x = r + (y \times q))
\end{array}
$$

- Example properties of the program DIV.
  - on every execution if AtEnd is true then Invariant is true and YleqR is not true
  - on every execution there is a state where AtEnd is true
  - on any execution if there exists a state where YleqR is true then there is also a state where InLoop is true
- Compare these with what is expressible in Hoare logic
  - execution: a path starting from a state satisfying AtStart

Recall `JM1`: a non-deterministic program example

Thread 1	Thread 2
`0:  IF LOCK=0 THEN LOCK:=1;`	`0:  IF LOCK=0 THEN LOCK:=1;`
`1:  X:=1;`	`1:  X:=2;`
`2:  IF LOCK=1 THEN LOCK:=0;`	`2:  IF LOCK=1 THEN LOCK:=0;`
`3:`	`3:`

$$S_{\texttt{JM1}} \;=\; [0..3] \times [0..3] \times \mathbb{Z} \times \mathbb{Z}$$

$$\begin{array}{lllll}
\forall pc_1\ pc_2\ lock\ x.\ & R_{\texttt{JM1}} & (0, pc_2, 0, x) & (1, pc_2, 1, x) & \wedge \\
& R_{\texttt{JM1}} & (1, pc_2, lock, x) & (2, pc_2, lock, 1) & \wedge \\
& R_{\texttt{JM1}} & (2, pc_2, 1, x) & (3, pc_2, 0, x) & \wedge \\
& R_{\texttt{JM1}} & (pc_1, 0, 0, x) & (pc_1, 1, 1, x) & \wedge \\
& R_{\texttt{JM1}} & (pc_1, 1, lock, x) & (pc_1, 2, lock, 2) & \wedge \\
& R_{\texttt{JM1}} & (pc_1, 2, 1, x) & (pc_1, 3, 0, x) &
\end{array}$$

- An atomic property:
  - $\texttt{NotAt11}(pc_1, pc_2, lock, x) = \neg((pc_1 = 1) \wedge (pc_2 = 1))$
- A non-atomic property:
  - all states reachable from $(0, 0, 0, 0)$ satisfy `NotAt11`
  - this is an example of a reachability property

State satisfying `NotAt11` unreachable from $(0,0,0,0)$

```
Thread 1                            Thread 2
0:  IF LOCK=0 THEN LOCK:=1;         0:  IF LOCK=0 THEN LOCK:=1;
1:  X:=1;                           1:  X:=2;
2:  IF LOCK=1 THEN LOCK:=0;         2:  IF LOCK=1 THEN LOCK:=0;
3:                                  3:
```

$R_{\mathrm{JM1}}\ (0, pc_2, 0, x)\quad (1, pc_2, 1, x)$
$R_{\mathrm{JM1}}\ (1, pc_2, lock, x)\quad (2, pc_2, lock, 1)$
$R_{\mathrm{JM1}}\ (2, pc_2, 1, x)\quad (3, pc_2, 0, x)$

$R_{\mathrm{JM1}}\ (pc_1, 0, 0, x)\quad (pc_1, 1, 1, x)$
$R_{\mathrm{JM1}}\ (pc_1, 1, lock, x)\quad (pc_1, 2, lock, 2)$
$R_{\mathrm{JM1}}\ (pc_1, 2, 1, x)\quad (pc_1, 3, 0, x)$

- $\mathtt{NotAt11}(pc_1, pc_2, lock, x) = \neg((pc_1 = 1) \wedge (pc_2 = 1))$
- Can only reach $pc_1 = 1 \ \wedge\ pc_2 = 1$ via:

  $R_{\mathrm{JM1}}\ (0, pc_2, 0, x)\quad (1, pc_2, 1, x)$ i.e. a step $R_{\mathrm{JM1}}\ (0,1,0,x)\quad (1,1,1,x)$
  $R_{\mathrm{JM1}}\ (pc_1, 0, 0, x)\quad (pc_1, 1, 1, x)$ i.e. a step $R_{\mathrm{JM1}}\ (1,0,0,x)\quad (1,1,1,x)$
- But:

  $R_{\mathrm{JM1}}\ (pc_1, pc_2, lock, x)\ (pc_1', pc_2', lock', x')\ \wedge\ pc_1'{=}0\ \wedge\ pc_2'{=}1\ \Rightarrow\ lock'{=}1$
  $\wedge$
  $R_{\mathrm{JM1}}\ (pc_1, pc_2, lock, x)\ (pc_1', pc_2', lock', x')\ \wedge\ pc_1'{=}1\ \wedge\ pc_2'{=}0\ \Rightarrow\ lock'{=}1$
- So can never reach $(0,1,0,x)$ or $(1,0,0,x)$
- So can't reach $(1,1,1,x)$, hence never $(pc_1 = 1) \wedge (pc_2 = 1)$
- Hence all states reachable from $(0,0,0,0)$ satisfy `NotAt11`

Reachability

- $R\ s\ s'$ means s' reachable from s in one step
- $R^n\ s\ s'$ means s' reachable from s in n steps

  $$\begin{array}{lcl} R^0\ s\ s' & = & (s = s') \\ R^{n+1}\ s\ s' & = & \exists s''.\ R\ s\ s'' \wedge R^n\ s''\ s' \end{array}$$

- $R^*\ s\ s'$ means s' reachable from s in finite steps

  $$R^*\ s\ s' = \exists n.\ R^n\ s\ s'$$

- Note: $R^*\ s\ s' \Leftrightarrow \exists \pi\ n.\ \mathrm{Path}\ R\ s\ \pi \wedge (s' = \pi(n))$
- The set of states reachable from s is $\{s' \mid R^*\ s\ s'\}$
- Verification problem: all states reachable from s satisfy p
  - verify truth of $\forall s'.\ R^*\ s\ s' \Rightarrow p(s')$
  - e.g. all states reachable from $(0,0,0,0)$ satisfy $\mathtt{NotAt11}$
  - i.e. $\forall s'.\ R^*_{\mathtt{JM1}}\ (0,0,0,0)\ s' \Rightarrow \mathtt{NotAt11}(s')$

Models and model checking

- Assume a model (S, R)
- Assume also a set $S_0 \subseteq S$ of initial states
- Assume also a set AP of atomic properties
  - allows different models to have same atomic properties
- Assume a labelling function $L : S \to \mathcal{P}(AP)$
  - $p \in L(s)$ means "s labelled with p" or "p true of s"
  - previously properties were functions $p : S \to \mathbb{B}$
  - now $p \in AP$ is distinguished from $\lambda s.\ p \in L(s)$
  - assume $\mathtt{T}, \mathtt{F} \in AP$ with forall s: $\mathtt{T} \in L(s)$ and $\mathtt{F} \notin L(s)$
- A *Kripke structure* is a tuple (S, S_0, R, L)
  - often the term "model" is used for a Kripke structure
  - i.e. a model is (S, S_0, R, L) rather than just (S, R)
- Model checking computes whether $(S, S_0, R, L) \models \phi$
  - ϕ is a property expressed in a property language
  - informally $M \models \phi$ means "wff ϕ is true in model M"

Minimal property language: ϕ is $\mathbf{AG}\,p$ where $p \in AP$

- Consider properties ϕ of form $\mathbf{AG}\,p$ where $p \in AP$
  - "**AG**" stands for "Always Globally"
  - from CTL (same meaning, more elaborately expressed)
- Assume $M = (S, S_0, R, L)$
- Reachable states of M are $\{s' \mid \exists s \in S_0.\ R^*\ s\ s'\}$
  - i.e. the set of states reachable from an initial state
- Define $\mathrm{Reachable}\ M = \{s' \mid \exists s \in S_0.\ R^*\ s\ s'\}$
- $M \models \mathbf{AG}\,p$ means p true of all reachable states of M
- If $M = (S, S_0, R, L)$ then $M \models \phi$ formally defined by:

$$\boxed{M \models \mathbf{AG}\,p \Leftrightarrow \forall s'.\ s' \in \mathrm{Reachable}\ M \Rightarrow p \in L(s')}$$

Model checking $M \models \mathbf{AG}\, p$

- $M \models \mathbf{AG}\, p \Leftrightarrow \forall s'.\ s' \in \text{Reachable } M \Rightarrow p \in L(s')$
  $\Leftrightarrow \text{Reachable } M \subseteq \{s' \mid p \in L(s')\}$

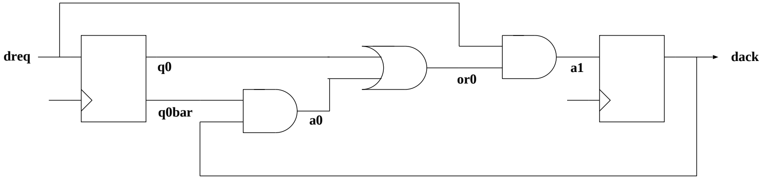
  checked by:
  - first computing Reachable M
  - then checking p true of all its members
- Let $\mathcal{S}$ abbreviate $\{s' \mid \exists s \in S_0.\ R^*\ s\ s'\}$ (i.e. Reachable M)
- Compute $\mathcal{S}$ iteratively: $\mathcal{S} = \mathcal{S}_0 \cup \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_n \cup \cdots$
  - i.e. $\mathcal{S} = \bigcup_{n=0}^{\infty} \mathcal{S}_n$
  - where: $\mathcal{S}_0 = S_0$ (set of initial states)
  - and inductively: $\mathcal{S}_{n+1} = \mathcal{S}_n \cup \{s' \mid \exists s \in \mathcal{S}_n \wedge R\ s\ s'\}$
- Clearly $\mathcal{S}_0 \subseteq \mathcal{S}_1 \subseteq \cdots \subseteq \mathcal{S}_n \subseteq \cdots$
- Hence if $\mathcal{S}_{\mathrm{m}} = \mathcal{S}_{\mathrm{m}+1}$ then $\mathcal{S} = \mathcal{S}_{\mathrm{m}}$
- Algorithm: compute $\mathcal{S}_0, \mathcal{S}_1, \ldots,$ until no change;
  check all members of computed set labelled with p

compute $\mathcal{S}_0$, $\mathcal{S}_1$, . . . , until no change;
check p holds of all members of computed set

- Does the algorithm terminate?
  - yes, if set of states is finite, because then no infinite chains: $\mathcal{S}_0 \subset \mathcal{S}_1 \subset \cdots \subset \mathcal{S}_n \subset \cdots$
- How to represent $\mathcal{S}_0$, $\mathcal{S}_1$, . . . ?
  - explicitly (e.g. lists or something more clever)
  - symbolic expression
- Huge literature on calculating set of reachable states

Example: RCV

- Recall the handshake circuit:



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

  $M = (S_{\mathrm{RCV}}, \{(1, 1, 1)\}, R_{\mathrm{RCV}}, L_{\mathrm{RCV}})$

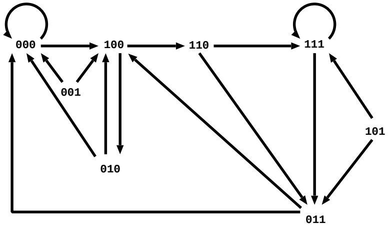
  and

  $$R_{\mathrm{RCV}}\ (dreq, q0, dack)\ (dreq', q0', dack') = \\ (q0' = dreq) \wedge (dack' = (dreq \wedge (q0 \vee dack)))$$

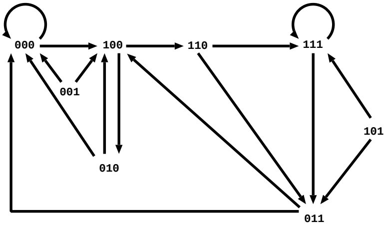
- AP and labelling function L_{RCV} discussed later

$\mathtt{RCV}$ state transition diagram

- Possible states for $\mathtt{RCV}$:
  $\{000, 001, 010, 011, 100, 101, 110, 111\}$
  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:



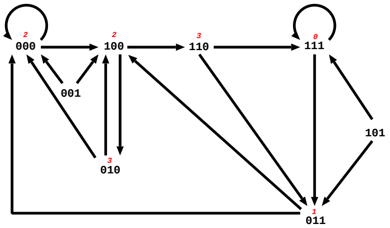
Computing Reachable M_{RCV}



- 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 Reachable M_{RCV} (continued)



- Compute:

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

- Hence Reachable $M_{\mathrm{RCV}} = \{111, 011, 000, 100, 010, 110\}$

Model checking $M_{\mathrm{RCV}} \models \mathbf{AG}\, p$

- $M = (S_{\mathrm{RCV}}, \{111\}, R_{\mathrm{RCV}}, L_{\mathrm{RCV}})$

- To check $M_{\mathrm{RCV}} \models \mathbf{AG}\, p$
  - compute Reachable $M_{\mathrm{RCV}} = \{111, 011, 000, 100, 010, 110\}$
  - check Reachable $M_{\mathrm{RCV}} \subseteq \{s \mid p \in L_{\mathrm{RCV}}(s)\}$
  - i.e. check if $s \in$ Reachable M_{RCV} then $p \in L_{\mathrm{RCV}}(s)$, i.e.:

    $p \in L_{\mathrm{RCV}}(111) \wedge$
    $p \in L_{\mathrm{RCV}}(011) \wedge$
    $p \in L_{\mathrm{RCV}}(000) \wedge$
    $p \in L_{\mathrm{RCV}}(100) \wedge$
    $p \in L_{\mathrm{RCV}}(010) \wedge$
    $p \in L_{\mathrm{RCV}}(110)$

- Example
  - if $AP = \{\mathtt{A}, \mathtt{B}\}$
  - and $L_{\mathrm{RCV}}(s) \;=\; \mathit{if}\ s \in \{001, 101\}\ \mathit{then}\ \{\mathtt{A}\}\ \mathit{else}\ \{\mathtt{B}\}$
  - then $M_{\mathrm{RCV}} \models \mathbf{AG}\, \mathtt{A}$ is not true, but $M_{\mathrm{RCV}} \models \mathbf{AG}\, \mathtt{B}$ is true

Symbolic Boolean model checking of reachability

- Assume states are n-tuples of Booleans $(b_1, \ldots, b_n)$
  - $b_i \in \mathbb{B} = \{true, false\}$ ($= \{1, 0\}$)
  - $S = \mathbb{B}^n$, so S is finite: 2^n states
- Assume n distinct Boolean variables: v_1,...,v_n
  - e.g. if $n = 3$ then could have $v_1 = \mathtt{x}$, $v_2 = \mathtt{y}$, $v_3 = \mathtt{z}$
- Boolean formula $f(v_1, \ldots, v_n)$ represents a subset of S
  - $f(v_1, \ldots, v_n)$ only contains variables v_1,...,v_n
  - $f(b_1, \ldots, b_n)$ denotes result of substituting b_i for v_i
  - $f(v_1, \ldots, v_n)$ determines $\{(b_1, \ldots, b_n) \mid f(b_1, \ldots, b_n) \Leftrightarrow true\}$
- Example $\neg(\mathtt{x} = \mathtt{y})$ represents $\{(true, false), (false, true)\}$
- Transition relations also represented by Boolean formulae
  - e.g. R_{RCV} represented by:

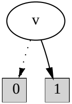
    $(q0' = dreq) \land (dack' = (dreq \land (q0 \lor (\neg q0 \land dack))))$

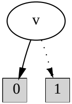
Symbolically represent Boolean formulae as BDDs

- Key features of Binary Decision Diagrams (BDDs):
  - canonical (given a variable ordering)
  - efficient to manipulate
- Variables:

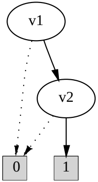
```
v    =   if v then 1 else 0
¬v   =   if v then 0 else 1
```

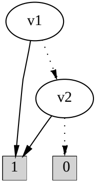
- Example: BDDs of variable `v` and `¬v`





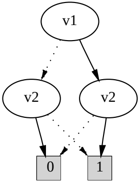
- Example: BDDs of `v1` $\wedge$ `v2` and `v1` $\vee$ `v2`



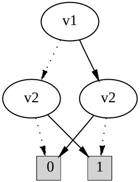


More BDD examples

- BDD of $\texttt{v1} = \texttt{v2}$



- BDD of $\texttt{v1} \neq \texttt{v2}$

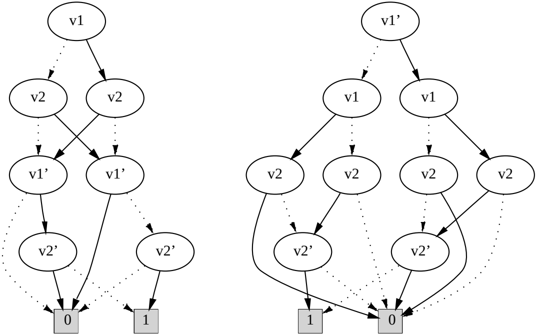


BDD of a transition relation

- BDDs of

  $$(\mathtt{v1}' = (\mathtt{v1} = \mathtt{v2})) \wedge (\mathtt{v2}' = (\mathtt{v1} \neq \mathtt{v2}))$$

  with two different variable orderings



- **Exercise:** draw BDD of $\mathcal{R}_{\mathrm{RCV}}$

Standard BDD operations

- If formulae f_1, f_2 represents sets S_1, S_2, respectively then $f_1 \wedge f_2$, $f_1 \vee f_2$ represent $S_1 \cap S_2$, $S_1 \cup S_2$ respectively
- Standard algorithms compute Boolean operation on BDDs
- Abbreviate $(v_1, \ldots, v_n)$ to $\vec{v}$
- If $f(\vec{v})$ represents S and $g(\vec{v}, \vec{v}')$ represents $\{(\vec{v}, \vec{v}') \mid R\ \vec{v}\ \vec{v}')\}$ then $\exists \vec{u}.\ f(\vec{u}) \wedge g(\vec{u}, \vec{v})$ represents $\{\vec{v} \mid \exists \vec{u}.\ \vec{u} \in S \wedge R\ \vec{u}\ \vec{v}\}$
- Can compute BDD of $\exists \vec{u}.\ h(\vec{u}, \vec{v})$ from BDD of $h(\vec{u}, \vec{v})$
  - e.g. BDD of $\exists v_1.\ h(v_1, v_2)$ is BDD of $h(\mathtt{T}, v_2) \vee h(\mathtt{F}, v_2)$
- From BDD of formula $f(v_1, \ldots, v_n)$ can compute $b_1, \ldots, b_n$ such that if $v_1 = b_1, \ldots, v_n = b_n$ then $f(b_1, \ldots, b_n) \Leftrightarrow \mathit{true}$
  - $b_1, \ldots, b_n$ is a satisfying assignment (SAT problem)
  - used for counterexample generation (see later)