T-79.298 Postgraduate Course in Digital Systems Science

Model Checking Algorithms and Reactive Systems

Satu Virtanen, satu@cs.hut.fi

12 November 2001

- *initial* satisfiability : whether $(U, \mathcal{I}, w_0) \models \varphi$
- *universal* satisfiability: whether $(U, \mathcal{I}) \models \varphi$, i.e. $\forall w_0 \in U : (U, \mathcal{I}, w_0) \models \varphi$
- *global* model checking algorithms iterate on the structure of φ and traverse the entire $\mathcal{M}$ per each step
- *local* model checking algorithms extend the portion of $\mathcal{M}$ to be examined per each round in an iterative manner, examining the entire φ per each step

Branching time logics

$$\mathbf{ML} ::= \mathcal{P} \mid \perp \mid (\mathbf{ML} \rightarrow \mathbf{ML}) \mid \langle \mathcal{R} \rangle \mathbf{ML}$$

The validity of a multimodal formula φ is evaluated according to the following rule set.

(i) $\mathcal{M} \models \mathtt{p}$ iff $w_0 \in \mathcal{I}(\mathtt{p})$

(ii) $\mathcal{M} \not\models \perp$

(iii) $\mathcal{M} \models (\varphi \rightarrow \psi)$ iff $\mathcal{M}\varphi$ implies $\mathcal{M} \models \psi$

(iv) $\mathcal{M} \models \langle R \rangle \varphi$ iff there exists a $w_1 \in U$ such that $(w_0, w_1) \in \mathcal{I}(R)$ and $(U, \mathcal{I}, w_1) \models \varphi$

The set of states satistyfing φ is denoted with $\varphi^{\mathcal{F}}$ and calculated recursively:

(i) for an *atomic proposition* $\mathtt{p}$, $\mathtt{p}^{\mathcal{F}} \triangleq \mathcal{I}(\mathtt{p})$

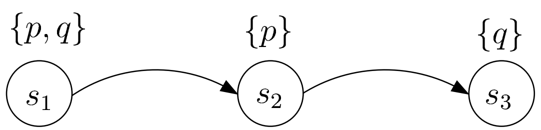
(ii) $\perp^{\mathcal{F}} \triangleq \emptyset$

(iii) $(\phi \rightarrow \psi)^{\mathcal{F}} \triangleq (U \setminus \phi^{\mathcal{F}}) \cup \psi^{\mathcal{F}}$

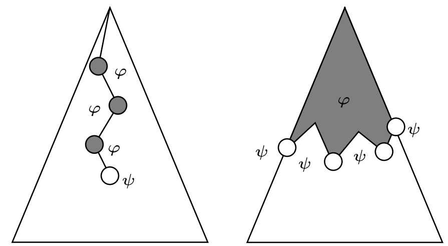
(iv) $(\langle R \rangle \psi)^{\mathcal{F}} \triangleq \{w \in U \mid \exists w' \in \psi^{\mathcal{F}}, (w, w') \in \mathcal{I}(R)\}$

Computational complexity: $\mathcal{O}(|\mathcal{M}| \times |\varphi|)$

Verifying a modal logic formula $\varphi = (\mathtt{q} \rightarrow \langle R \rangle \mathtt{p})$



CTL $::= \mathcal{P} \mid \bot \mid (\mathbf{CTL} \rightarrow \mathbf{CTL}) \mid \mathbf{E}(\mathbf{CTL}\ \mathbf{U}^{+}\mathbf{CTL}) \mid \mathbf{A}(\mathbf{CTL}\ \mathbf{U}^{+}\mathbf{CTL})$



$\mathbf{E}(\varphi \mathbf{U}^{+} \psi)$ $\mathbf{A}(\varphi \mathbf{U}^{+} \psi)$

$w_0 \models (\phi\, \mathbf{U}^{+} \psi)$ iff $\exists w_1 \in U$ such that $w_0 < w_1$ and $w_1 \models \psi$, also $\forall w_2 \in U$ such that $w_0 < w_2 < w1$, applies that $w_2 \models \phi$.

$$\mathbf{E}(\psi_2\, \mathbf{U}^{+} \psi_1) \leftrightarrow \mathbf{EX}(\psi_1 \vee (\psi_2 \wedge \mathbf{E}(\psi_2\, \mathbf{U}^{+} \psi_1)))$$

$$\mathbf{A}(\psi_2\, \mathbf{U}^{+} \psi_1) \leftrightarrow \mathbf{AX}(\psi_1 \vee (\psi_2 \wedge \mathbf{A}(\psi_2\, \mathbf{U}^{+} \psi_1)))$$

procedure `CTLcheck` (model $\mathcal{M} = (U, \mathcal{I}, w_0)$, formula φ) {

if $w_0 \in$ `eval`(φ)
  `print` "φ *is* satisfied at w_0 of the frame $\mathcal{F} = (U, \mathcal{I})$"
else
  `print` "φ *is not* satisfied at w_0 of the frame"

function `succ` (state w) : set of states
  return $\{w' \mid (w, w') \in \mathcal{I}(\prec)\}$ // RETURNS STATES REACHABLE BY "$\prec$"

Computational complexity: $\mathcal{O}(|\varphi| \cdot |U|^3)$

function `eval` (formula ϕ) : set of states

  case ϕ **of**

    p: **return** $\mathcal{I}(p)$

    $\bot$: **return** $\emptyset$

    $(\psi_1 \to \psi_2)$: **return** $((U \setminus \texttt{eval}(\psi_1)) \cup \texttt{eval}(\psi_2))$

    $\mathbf{E}(\psi_2\,\mathbf{U}^{+}\psi_1)$: $E_1 := \texttt{eval}(\psi_1)$, $E_2 := \texttt{eval}(\psi_2)$, $E := \emptyset$

      repeat until stabilization // UNTIL E DOES NOT GROW

        $E := E \cup \{w \mid (\texttt{succ}(w) \cap (E_1 \cup (E_2 \cap E))) \neq \emptyset\}$

      return E

    $\mathbf{A}(\psi_2\,\mathbf{U}^{+}\psi_1)$: $E_1 := \texttt{eval}(\psi_1)$, $E_2 := \texttt{eval}(\psi_2)$, $E := \emptyset$

      repeat until stabilization // UNTIL E DOES NOT GROW

        $E := E \cup \{w \mid \emptyset \neq \texttt{succ}(w) \subseteq (E_1 \cup (E_2 \cap E))\}$

      return E

Inverse reachability problem

Computational complexity: $\mathcal{O}(|E|)$.

function `inverseReachability`(set of points T) : set of points

  <set> $S := \emptyset$ // THE RESULTING SET OF NODES IS INITIALLY EMPTY

  <set> $T' = T$ // A WORKING SET FOR THE ALGORITHM

  while $T' \neq \emptyset$ **do**

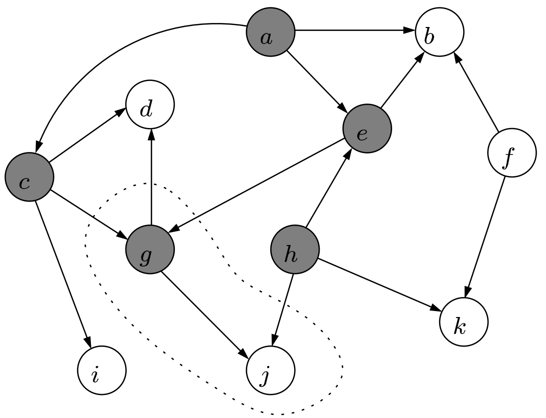
    $T' := \texttt{predecessors}(T') \setminus S$

    $S := S \cup T'$

  return S;

function `predecessors`(point w) : set of points

  return $\{w' \mid (w', w) \in \mathcal{I}(\prec)\}$



A procedure for marking where $\mathbf{EG}^{+}\varphi$ holds

(i) exclude all states in which φ does not hold

(ii) mark the remaining set with $V_\varphi \subseteq U$

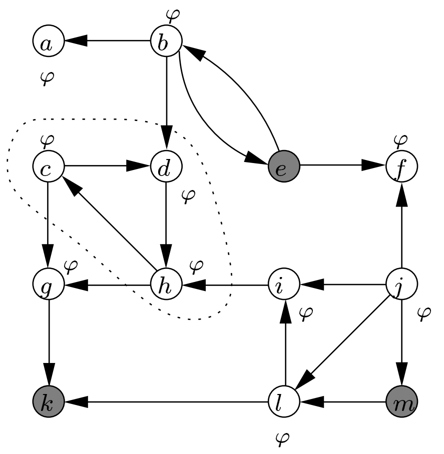
(iii) mark all states $w \in U$ from which a state $w' \in V_\varphi$ without any successors at all that can be reached traversing only through states of V_φ

(iv) find the maximal strongly connected components (SCCs) of V_φ

(v) mark all states that are members of SCCs

(vi) mark all states from which a non-trivial SCC of V_φ can be reached by a path in V_φ

$\psi^{\mathcal{F}} = \{b, c, d, e, h, i, j, l, m\}$ for $\psi = \mathbf{EG}^{+}\varphi$ for the $\mathcal{M}$ below:



Validity for linear time logics

- the set of *maximal sequences* generated from $\mathcal{M} = (U, \mathcal{I}, w_0)$ starting at the w_0 satisfies φ
- whether $\neg\varphi$ is valid for even one sequence

A simplified modal logic:

$$\mathbf{ML} ::= \mathcal{P} \,|\, \bot \,|\, (\mathbf{ML} \rightarrow \mathbf{ML}) \,|\, \langle R \rangle \mathbf{ML}$$

For given $\mathcal{M}$ and φ: can a maximal sequence be generated from $\mathcal{M}$ starting at w_0 such that $\neg\varphi$ is satisfied at w_0?

A set m of subformulas of φ is said to be *maximal* if for $\psi \in SF(\varphi)$: for each element, either the element or its negation is included in the set m, i.e. $\{\psi \,|\, \psi \in m\} \cup \{\neg\psi \,|\, \psi \notin m\}$. Such a set is denoted as $m \in SF(\varphi)$

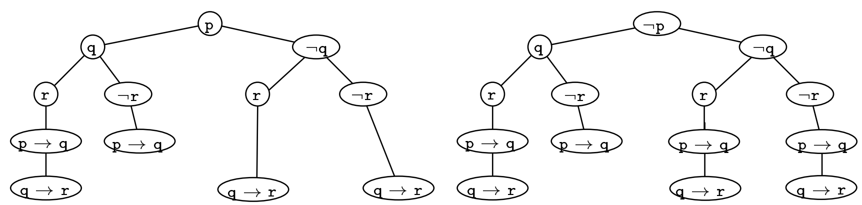
A set $m \subseteq SF(\varphi)$ is said to be *propositionally consistent* if both of the requirements below hold:

(i) $\bot \notin m$

(ii) if $(\psi_1 \rightarrow \psi_2) \in SF(\varphi)$, then the it must hold that $(\psi_1 \rightarrow \psi_2) \in m$ iff $\psi_1 \notin w$ or $\psi_2 \in m$

Finding propositionally consistent sets

$\varphi = (\mathtt{q} \rightarrow \langle R \rangle (\mathtt{p} \rightarrow \mathtt{q}))$, $SF(\varphi) = \{\mathtt{q}, \langle R \rangle (\mathtt{p} \rightarrow \mathtt{q}), (\mathtt{p} \rightarrow \mathtt{q}), \mathtt{p}, \varphi\}$



Maximal propositionally consistent sets

$$
\begin{array}{lll}
m & \in & \{ \quad \{\texttt{p}, \texttt{q}, \texttt{r}, (\texttt{p} \rightarrow \texttt{q}), (\texttt{q} \rightarrow \texttt{r})\}, \\
& & \quad\ \{\texttt{p}, \texttt{q}, \neg\texttt{r}, (\texttt{p} \rightarrow \texttt{q})\} \\
& & \quad\ \{\texttt{p}, \neg\texttt{q}, \texttt{r}, (\texttt{q} \rightarrow \texttt{r})\} \\
& & \quad\ \{\texttt{p}, \neg\texttt{q}, \neg\texttt{r}, (\texttt{q} \rightarrow \texttt{r})\} \\
& & \quad\ \{\neg\texttt{p}, \texttt{q}, \texttt{r}, (\texttt{p} \rightarrow \texttt{q}), (\texttt{q} \rightarrow \texttt{r})\}, \\
& & \quad\ \{\neg\texttt{p}, \texttt{q}, \neg\texttt{r}, (\texttt{p} \rightarrow \texttt{q})\} \\
& & \quad\ \{\neg\texttt{p}, \neg\texttt{q}, \texttt{r}, (\texttt{p} \rightarrow \texttt{q}), (\texttt{q} \rightarrow \texttt{r})\}, \\
& & \quad\ \{\neg\texttt{p}, \neg\texttt{q}, \neg\texttt{r}, (\texttt{p} \rightarrow \texttt{q}), (\texttt{q} \rightarrow \texttt{r})\} \quad \}
\end{array}
$$

Shorthand: $(\texttt{q} \rightarrow \texttt{r}) \triangleq \varphi$

Constructing a forest of admissible atoms

- *atom*: any pair (w, m), where $w \in U$ and $m \subseteq SF(\varphi)$
- *admissible atom*: w and m *agree on the interpretation of propositions*
- *initial atom*: (w_0, m_0) where $\varphi \in m_0$
- *open leaf*: $\alpha = (w, m)$ with no children, is called *open* that has no formulas in m that begins with $\langle R \rangle$ in m
- *closed leaf*: other atoms in the forest
- *accepting path*: a path in the forest that is either infinite or ends in an open leaf (a counterexample to φ)

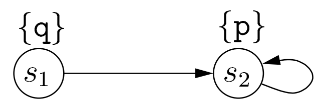
$X_R((w, m), (w', m'))$ holds between admissible atoms iff

(i) $(w, w') \in \mathcal{I}(R)$

(ii) if $\langle R \rangle \psi \in SF(\varphi)$ and $\psi \in m'$, then $\langle R \rangle \psi \in m$

(iii) if $\langle R \rangle \psi \in m$, then $\psi \in m'$

(iv) some $\langle R \rangle \psi \in m$



Admissible atoms for $\varphi = (\texttt{q} \rightarrow \langle R \rangle (\texttt{p} \rightarrow \texttt{q}))$ and above $\mathcal{M}$:

α	w	$m \in SF(\varphi)$
α_1	s_1	$\{\texttt{p}, \neg\texttt{q}, \langle R \rangle (\texttt{p} \rightarrow \texttt{q}), \varphi\}$
α_2	s_1	$\{\texttt{p}, \neg\texttt{q}, \varphi\}$
α_3	s_2	$\{\neg\texttt{p}, \texttt{q}, \langle R \rangle (\texttt{p} \rightarrow \texttt{q}), (\texttt{p} \rightarrow \texttt{q}), \varphi\}$
α_4	s_2	$\{\neg\texttt{p}, \texttt{q}, (\texttt{p} \rightarrow \texttt{q})\}$

α_1 and α_2 are initial atoms, X_R contains (α_1, α_3) and (α_3, α_4).

LTL

Formulas in positive conjunctive normal form:

$$\mathbf{F}^*\mathtt{p}_1 \wedge \ldots \wedge \mathbf{F}^*\mathtt{p}_n \wedge \mathbf{G}^*(\mathtt{p}_1 \rightarrow \mathbf{XG}^*\neg\mathtt{p}_1) \wedge \ldots \mathbf{G}^*(\mathtt{p}_n \rightarrow \mathbf{XG}^*\neg\mathtt{p}_n)$$

Definition of $\mathcal{M} = (U, \mathcal{I}, w_0)$:

- the set of states $U = V \cup \{s, t\}$, where $s, t \notin V$
- $w_0 = s$
- $\mathcal{I}(R) = E \cup \{(s, v_i) \,|\, v_i \in V\} \cup \{(v_i, t) \,|\, v_i \in V\} \cup \{(t, t)\}$
- $\mathcal{I}(\mathtt{p})$ for all atomic propositions $\mathtt{p} \in \mathcal{P}$ is such that
  - $v_i \in \mathcal{I}(\mathtt{p}_i)$ iff $1 \leq i \leq |V|$
  - $v_i \notin \mathcal{I}(\mathtt{p}_j)$ iff $1 \leq i, j \leq |V|, i \neq j$
  - $\forall 1 \leq i \leq |V| : s \notin \mathcal{I}(\mathtt{p}_i)$
  - $\forall 1 \leq i \leq |V| : t \notin \mathcal{I}(\mathtt{p}_i)$

The *closure* of a formula φ, $CL(\varphi)$, is defined as the minimal set of formulas containing φ for which

(i) $\neg\psi \in CL(\varphi)$ if and only if $\psi \in CL(\varphi)$

(ii) if $\psi_1 \vee \psi_2 \in CL(\varphi)$, then both $\psi_1, \psi_2 \in CL(\varphi)$

(iii) if $\mathbf{X}\psi \in CL(\varphi)$, then $\psi \in CL(\varphi)$

(iv) if $\neg\mathbf{X}\psi \in CL(\varphi)$, then $\mathbf{X}\neg\psi \in CL(\varphi)$

(v) if $(\psi_2\,\mathbf{U}^*\psi_1) \in CL(\varphi)$, then all $\psi_1, \psi_2, \mathbf{X}(\psi_2\,\mathbf{U}^*\psi_1) \in CL(\varphi)$

Atoms $\alpha = (w, c)$ where $w \in U$ and c as defined below:

- $\forall \mathtt{p} \in \mathcal{P} : \mathtt{p} \in c_i$ if and only if $w_i \in \mathcal{I}(\mathtt{p})$
- $\forall \psi \in CL(\varphi) : \psi \in c_i$ if and only if $\neg\psi \notin c_i$
- $\forall (\psi_1 \vee \psi_2) \in CL(\varphi) : (\psi_1 \vee \psi_2) \in c_i$ if and only if either $\psi_1 \in c_i$ or $\psi_2 \in c_i$
- $\forall \neg\mathbf{X}\psi \in CL(\varphi) : \neg\mathbf{X}\psi \in c_i$ if and only if $\mathbf{X}\neg\psi \in c_i$
- $\forall (\psi_2\,\mathbf{U}^*\psi_1) \in CL(\varphi)$, $(\psi_2\,\mathbf{U}^*\psi_1) \in c_i$ if and only if either $\psi_1 \in c_i$ or both $\psi_2 \in c_i$ and $\mathbf{X}(\psi_2\,\mathbf{U}^*\psi_1) \in c_i$

$(\alpha, \alpha') \in E$ for $\alpha = (w, c)$ and $\alpha' = (w', c')Q$ iff

- $(\alpha, \alpha') \in E$ if and only if $(w, w') \in \mathcal{I}(R)$ and
- $\forall \mathbf{X}\psi \in CL(\varphi) : \mathbf{X}\psi \in c$ if and only if $\psi \in c'$

$\mathcal{M}, w_0 \models \mathbf{E}\,\varphi$ iff there exists an eventuality sequence starting at an initial atom $\alpha = (w_0, c_0)$.

An eventuality sequence starts from an atom $\alpha = (w, c)$ iff there there is a path in the atom graph G such that a self-fulfilling SCC can be reached by it from α.

Modified Tarjan's algorithm

```
procedure LTLcheck(model M, formula φ)
    <integer> recursionDepth := 0 // NUMBER OF RECURSIVE CALLS MADE
    <stack> SCC := ∅
    <hashtable> storage // FOR STORING THE TRAVERSED "PATH"
    <set> initial := {α | α is an initial atom of M and φ}
    for all α ∈ initial do dfs(α)
    print "φ is not satisfiable in M"
```

- exponential with respect to the number of $\mathbf{U}^{+}$formulas
- linear with respect to the size of $\mathcal{M}$, $\mathcal{O}(|U|^2)$
- generally: with past-temporal operators is complete in **PSPACE** with respect to $|\varphi|$ and **NLOGSPACE** in size of the model $\mathcal{M}$

```
procedure dfs(atom α) // DEPTH FIRST SEARCH
    if (storage does not contain α) // A "FRESH" ATOM
        <integer> currentDepth = recursionDepth
        recursionDepth++ // INCREMENT DEPTH BY ONE
        // INITIAL VALUE AT CURRENT RECURSION DEPTH
        storage[α] = currentDepth
        push(SCC, α) // TO THE TOP OF THE STACK
        <set> successorAtoms := children(α)
        for all (β ∈ successorAtoms) do
            dfs(β)
            // CHECK WHETHER β IS "ABOVE" α
            storage[α] = min(storage[α], storage[β])
        // IF NOTHING WAS ABOVE ⇒ α IS A ROOT OF AN SCC
        if (storage[α] = currentDepth)
            processRoot(α)
```

```
procedure processRoot (atom α)
    <set> required := ∅ // SET OF REQUIRED EVENTUALITIES
    <set> fulfilled := ∅ // SET OF FULFILLED EVENTUALITIES
    repeat
        β := pop(SCC)
        storage[β] := ∞ // PUT "BELOW" EVERYTHING ELSE
        required := required ∪ {ψ1 | (ψ2 U+ ψ1) ∈ β}
        fulfilled := fulfilled ∪ {ψ | ψ ∈ β}
    until (α = β) // ALL ELEMENTS IN THE SCC OF α ARE PROCESSED
    if (required ⊆ fulfilled) // THE SCC OF α IS SELF-FULFILLING
        print "φ is satisfiable in M"; exit
```

The function `children` returns the set of atoms accessible from the given atom in the constructed graph.

$$\mu\mathbf{TL} ::= \mathcal{P} \mid \mathcal{Q} \mid \bot \mid (\mu\mathbf{TL} \rightarrow \mu\mathbf{TL}) \mid \langle\mathcal{R}\rangle\mu\mathbf{TL} \mid \nu\mathcal{Q}\,\mu\mathbf{TL}$$

- the quantifier ν is a *restricted existential quantifier* on sets of points
- the quantifier μ is defined by ν as $\mu q\,\varphi \triangleq \neg\nu q\,\neg(\varphi\,\{q := \neg q\})$
- Knaster-Tarski:

$$(U, \mathcal{I}, w) \models \nu q \leftrightarrow w \in \bigcup\{Q \mid Q \subseteq \varphi^{\mathcal{F}}\{q := Q\}\}$$

$$(U, \mathcal{I}, w) \models \mu q \leftrightarrow w \in \bigcap\{Q \mid \varphi^{\mathcal{F}}\{q := Q\} \subseteq Q\}$$

If φ^i is union-continuous, the fixed points $\nu q\,\varphi$ and $\mu q\,\varphi$ can be obtained with the following limits:

$$\nu q\,\varphi = \lim_{i\to\infty} \varphi^i(\top)$$

$$\mu q\,\varphi = \lim_{i\to\infty} \varphi^i(\bot).$$

For a finite universe U, every monotonic function is also union-continuous:

$$\nu q\,\varphi = \lim_{i\le|U|} \varphi^i(\top)$$

$$\mu q\,\varphi = \lim_{i\le|U|} \varphi^i(\bot)$$

```
function eval (formula φ) : set of states
  case φ of
    p: return I(p) // ATOMIC PROPOSITION
    q: return v(q) // VALUATION OF A PROPOSITION VARIABLE
    ⊥: return ∅
    (ψ1 → ψ2): return ((U \ eval(ψ1)) ∪ eval(ψ2))
    ⟨R⟩ψ: return R^{-1}(eval(ψ))
    νq ψ : H1:= U
      repeat until stabilization // UNTIL H1 DOES NOT SHRINK
        H1 := eval(ψ{q := H1})
      return H1
    μq ψ : H2:= ∅
      repeat until stabilization // UNTIL H2 DOES NOT GROW
        H2 := eval(ψ{q := H2})
      return H2
```

- the cycles stabilize after at most $|U|$ rounds: $\mathcal{O}\left(|\varphi| \cdot |U|^{qd(\varphi)}\right)$
- $qd(\varphi)$ is the *nesting depth* of fixed-point operators in φ
- the computation of an inner fixed-point formula needs to be restarted for each iteration on the enclosing formula

Example: $\psi(q_1, q_2) = \mu q_1(\mathtt{p}_1 \wedge \mu q_2(\mathbf{X} q_1 \vee \mathbf{X} q_2 \vee \mathtt{p}_2))$.

For each iteration of q_1, the inner formula $\mu q_2(\mathbf{X} q_1 \vee \mathbf{X} q_2 \vee \mathtt{p}_2)$ is re-evaluated.

Local model checking for μTL

- a tableau method, which explores a portion of the model using depth-first search
- the nodes of the tableau are sequences of the form $\Delta, w \models \psi$, where $w \in U$ and ψ is a sub-formula of φ
- Δ is a list of definitions that contains a sequence of declarations $(q_1 = \psi_1, \ldots, q_n = \psi_n)$ where each proposition variable q_i appears only once and each ψ_i may only contain variables q_j such that $j < i$

Steps for decomposition of the formula φ into a tableau:

(i) a node $\Delta, w \models (\psi_1 \wedge \psi_2)$ has two children: $\Delta, w \models \psi_1$ and $\Delta, w \models \psi_2$

(ii) a node $\Delta, w \models (\psi_1 \vee \psi_2)$ has one child: either $\Delta, w \models \psi_1$ or $\Delta, w \models \psi_2$

(iii) a node $\Delta, w \models \langle R \rangle \psi$ has one child: $\Delta, w' \models \psi$

(iv) a node $\Delta, w \models [R]\psi$ has n children: $\Delta, w_1 \models \psi, \ldots, \Delta, w_n \models \psi$

(v) a node $\Delta, w \models \mu q\, \psi$ has one child: $\Delta', w \models \psi$

(vi) a node $\Delta, w \models \nu q\, \psi$ has one child: $\Delta', w \models \psi$

(vii) a node $\Delta, w \models q$ has one child: $\Delta, w \models \psi$

Restrictions for the construction:

$\vee$, $\wedge$, $\langle R \rangle$, $[R]$, μ and v are used as basic operations, negations are assumed to appear only in literals.

Each μ or ν quantification in the formula φ is expected to bind a different propositional variable.

- rule (iii) can only be applied when $w' \in R(w)$
- when applying rule (iv), it must hold that $R(w) = \{w_1, \ldots, w_n\}$
- for rules (v) and (vi), $\Delta' = \Delta \cup \{q = \psi\}$
- rule (vii) is only applicable when $(q = \psi) \in \Delta$ and no ancestor node is of the form $\Delta', w \models \psi$ with the same w and ψ

When no rule cannot be applied for any leaf, the tableau is said to be *maximal*.

A leaf $\Delta, w \models \psi$ of a maximal tableau is said to be *successful* if the following holds:

(1) $(\psi = \mathtt{p} \in \mathcal{P} \wedge w \in \mathcal{I}(\mathtt{p})) \vee (\psi = \neg\mathtt{p} \wedge w \notin \mathcal{I}(\mathtt{p}))$

(2) $\psi = q \in \mathcal{Q}, q \notin \Delta, w \in v(q)$, or $\psi = \neg q, q \notin \Delta, w \notin v(q)$, or

(3) rule (iv) produces no children, that is: $\psi = [R]\psi' \wedge R(w) = \emptyset$

(4) $\psi = q \in \mathcal{Q}$ and q was included in Δ by rule (vi)

Theorem: $w \in \varphi^{\mathcal{F}}$ iff there exists a successful tableau with root $\emptyset, w \models \varphi$.

Message passing

- processes send each other messages, where the messages and the order of sending them are specified by some protocol
- each process may *send* or *receive* a message.
- messages may be directed to specific *channels* with specified recipients or *broadcasted* to any process within the messaging space

In *synchronized* communication, the `send` and `receive` primitives are *blocking* operations, i.e. the sending process must wait for the recipient to react before proceeding further.

Synchronous communication: a process may not proceed until all the communication partners defined for the particular task have expressed willingness to participate in that action.

Asynchronous communication: the processes deciding themselves whether they wish to wait, usually by using some sort of buffer for messages that are not immediately reacted to

When non-determinism is required for the inter-process communication, *guards* that consists of a Boolean expression and communication statements may be placed on the messaging operations.

Communication by shared variables

When two or more processes interact by reading from and writing to a shared memory space, the may exchange information by using *shared variables* for separate issues on which to communicate. The semantic of possible values for each of the variables must be defined, similarly to defining a grammar or a protocol for the message passing paradigm.

It must be ensured that the accesses to the shared variables are *legal*: two processes should not be able to modify the value of a variable at the same time, and no process should read a value after some other process has reserved it for modification.

When the system modules are said to be *synchronized*, for each of the discrete time steps, each of the modules performs a task. The tasks are generally independent on each other, but it can also be stated that only when certain conditions apply, certain transitions may be taken by the processes.

One process may have to stay in an *idle loop*, performing essentially a `no-op` (stands for *no-operation*) until the conditions for taking the next actual step are reached.

In *asynchronous* computation, for each time step, some but not necessarily all modules advance. If performing a `no-op` is possible for every process in every state of the computation, then the synchronous and asynchronous computation modes coincide. The idle step is also called a "stutter".

Transition system (Σ, S, Δ, S_0)

- an *alphabet* Σ
- a non-empty and finite set of states S
- a subset $S_0 \subseteq S$ as *initial states*
- a *transition relation* $\Delta \subseteq S \times \Sigma \times S$ describes which states in S are reachable in one step from one another for the symbols of the alphabet Σ

Parallel transition system $T = (T_1, \dots, T_n)$

n-tuple of transition systems in which $\forall i, j$ such that $i < j$, applies that $S_i \cap S_j = \emptyset$.

- the *global alphabet* is the union of all alphabets, $\Sigma = \bigcup_{i=1}^{n} \Sigma_i$
- the *global state space* is defined as $S = S_1 \times \ldots \times S_n$
- the set of *global initial states* is defined as $S_0 = S_{10} \times \ldots \times S_{n0}$
- for the *global transition relation*, a transition $\delta = ((s_1, \dots, s_n), a, (s'_1, \dots, s'_n)) \in \Delta$ iff $\forall T_i$

  (i) if $a \in \Sigma_i$, then $(s_1, a, s'_1) \in \Delta_i$, and

  (ii) if $a \notin \Sigma_i$, then $s_i = s'_i$

Elementary Petri net $N = (P, T, F, m_0)$

- a finite set P of *places*
- a set of *transitions* T such that $P \cap T = \emptyset$
- a *flow relation* F that describes the relations of the places and the transitions: $F \subseteq (P \times T) \cup (T \times P)$
- an *initial marking* $m_0 \subseteq P$ imposed on the net
- any subset of P is a *marking* of the Petri net

A marking function $\mathcal{L}$ is a function from the set of places P to the set of P's subsets 2^P

- *preset* : $\bullet t \doteq \{p \,|\, (p,t) \in F\}$
- *postset* : $t\bullet \doteq \{p \,|\, (t,p) \in F\}$.

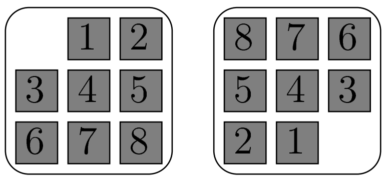
A transition is *enabled at a marking* $m \subseteq P$ if $\bullet t \subseteq m \wedge t \bullet \cap m \subseteq \bullet t$

A marking m' is said to be the *result of firing a transition* t from some marking m, if the transition t is enabled at m and the marking m' consists of the states $m' = (m \setminus \bullet t) \cup t\bullet$.

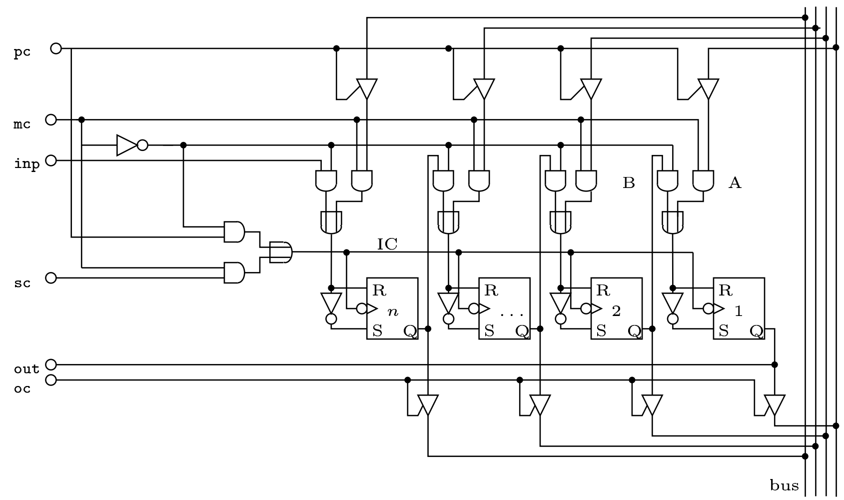
Shared variable program (V, D, T, s_0)

- a set of *program variables* $V = \{v_1, \dots, v_n\}$
- a *state space* $D = D_1 \times \ldots \times D_n$ in which each $D_i = \{d_{i1}, \dots, d_{im_i}\}$ is a *finite domain* of the corresponding variable v_i
- a *transition relation* T is defined as $T \subseteq D \times D$
- one *initial state* $s_0 = (d_{11}, \dots, d_{n1})$

A transition (s, s') is in T iff the proposition φ_T is a logical implication of the valuation $\mathcal{I}$, i.e. $\mathcal{I} \models \varphi_T$, where $s = (d_1, \dots, d_n)$, $s' = (d'_1, \dots, d'_n)$, $\mathcal{I}(v_i) = d_i$, and $\mathcal{I}(v'_i) = d'_i$



The puzzle as a shared variables program: a variable for each tile $i = 1, \ldots, (h \cdot v - 1)$ (a two-component vector $t_i = (v_{ih}, v_{iv})$) and one variable for the direction of the next move, transitions are swappings of a tile with the empty slot.



S	R	Q’	description
0	0	Q	maintain
1	0	1	set
0	1	0	reset
1	1	-	undefined

The undefined state can be modeled by making a non-deterministic choice between 0 and 1 for the value of the next state `Q’` internally. The output here changes only when the clock line changes from high to low.

The **correctness requirements** with n representing the width of the data bus are

$$\mathbf{AG}^*(\texttt{mc} \wedge \texttt{pc} \rightarrow \bigvee_{i=1}^{n} (\texttt{bus[i]} \leftrightarrow \mathbf{A}((\texttt{oc} \rightarrow \mathbf{AX}(\texttt{bus[i]}\,\mathbf{U}^{+}\texttt{ic})))$$

$$\mathbf{AG}^*(\neg\texttt{mc} \wedge \texttt{sc} \rightarrow \bigvee_{i=2}^{n} (\texttt{Q[i]} \leftrightarrow \mathbf{A}(\texttt{Q[i} - \texttt{1]}\,\mathbf{U}^{+}\texttt{ic}))),$$

`ic` is the result of the bit-wise operation $\texttt{ic} = ((\neg\texttt{mc} \wedge \texttt{pc}) \vee (\texttt{mc} \wedge \texttt{sc}))$.

```
MODULE main
VAR Q, bus:  array 1 ..  n of boolean;
   inp, mc, pc, sc, oc:  boolean; -- INPUT LINES
DEFINE out := Q[1]; ic := ((!mc & pc) | (mc & sc));
      -- FOR ALL OF THE n BITS INDEXED WITH 0,...,n-1:
      A[0] := mc & pc & bus[0]; B[0] := !mc & Q[1];
      R[0] := !(A[0] | B[0]); S[0] := !R[0];
ASSIGN next(Q[0]) := case ic:  case
            !S[0] & !R[0]:  Q[0]; -- MAINTAIN
            S[0] & !R[0]:  1; -- SET
            !S[0] & R[0]:  0; -- RESET
            S[0] & R[0]:  {0, 1}; esac; -- UNDEFINED
      next(bus[i]) := case oc:  Q[0]; !oc:  {0, 1}; esac;
FAIRNESS ic FAIRNESS oc
```

input	buffer	output	input	buffer	output
nil	$\langle\,\rangle$	**nil**		$\langle\,\rangle$	**nil**
x	$\langle\,\rangle$	**nil**	**nil**	$\langle\,\rangle$	x
nil	$\langle x_1,\ldots,x_v\rangle$	**nil**		$\langle x_1,\ldots,x_{v-1}\rangle$	x_v
x	$\langle x_1,\ldots,x_v\rangle$	**nil**	**nil**	$\langle x,x_1,\ldots,x_{v-1}\rangle$	x_v
nil	$\langle\,\rangle$	y		$\langle\,\rangle$	
x	$\langle\,\rangle$	y	**nil**	$\langle x\rangle$	
x	$\langle x_1,\ldots,x_v\rangle$	y	**nil**	$\langle x,x_1,\ldots,x_v\rangle$	
x	$\langle x_1,\ldots,x_v\rangle$	y	x	$\langle x_1,\ldots,x_n\rangle$	

A **bounded buffer** used by a set of communicating processes within the operating system of a Siemens cellular phone

- should be deadlock-free: **AG*****EF***`init`
- five processes and the operating system kernel process
- size of the state space per process is app. 10–20 states
- there are app. 50 types of messages
- processes scheduled by the operating system based on priorities
- storage and delivery of the messages is also done by the OS