ACTL, UCTL and their model checking

- In process algebras, a system is seen as a set of processes, and each process is characterised by the actions performed in the time by the process.

- Hence a model checker focusing on actions, rather than states, is able to address the whole word of systems commonly modelled with process algebras, and with languages derived by process algebras. These systems include concurrent and distributed systems, mobile systems, and cover the behavioural aspects of object-orented systems.

The ACTL logic: syntax

- **ACTL Syntax** --> $\phi ::= \text{true} \mid \sim\phi \mid \phi \wedge \phi \mid E\,\gamma \mid A\,\gamma$
  - **(state formulae)**

  $\gamma ::= X\chi\,\phi \mid X\tau\,\phi \mid \phi\ \chi\ U\ \chi\ \phi' \mid \phi\ \chi\ U\ \phi'$
  - **(path formulae)**

- **Action formulae** --> $\chi ::= \alpha \mid \sim\chi \mid \chi \wedge \chi \mid \chi \mid \chi$

- • **ACTL Semantics**
- The satisfation of an ACTL formula is inductively defined over labelled transition systems

Preliminary definitions

A labelled transition systems (LTS) is a 4-tuple $A=(S, s0, Act, \rightarrow)$, where:

S is a finite set of states; $s0$ is the initial state;

Act is a finite set of observable actions;

$\rightarrow \subseteq S \times Act \times S$ is the transition relation between states.

We denote by $s \text{ -}a\rightarrow s'$, $a \in Act$, the transition from the state s to the state s' by executing a; in particular, $s \text{ -}a\rightarrow s'$ indicates that a system in state s can perform a transition to state s' by executing the action a.

$D_a(s) = \{s' \mid s \text{ -}a\rightarrow s'\}$ set of successors of s

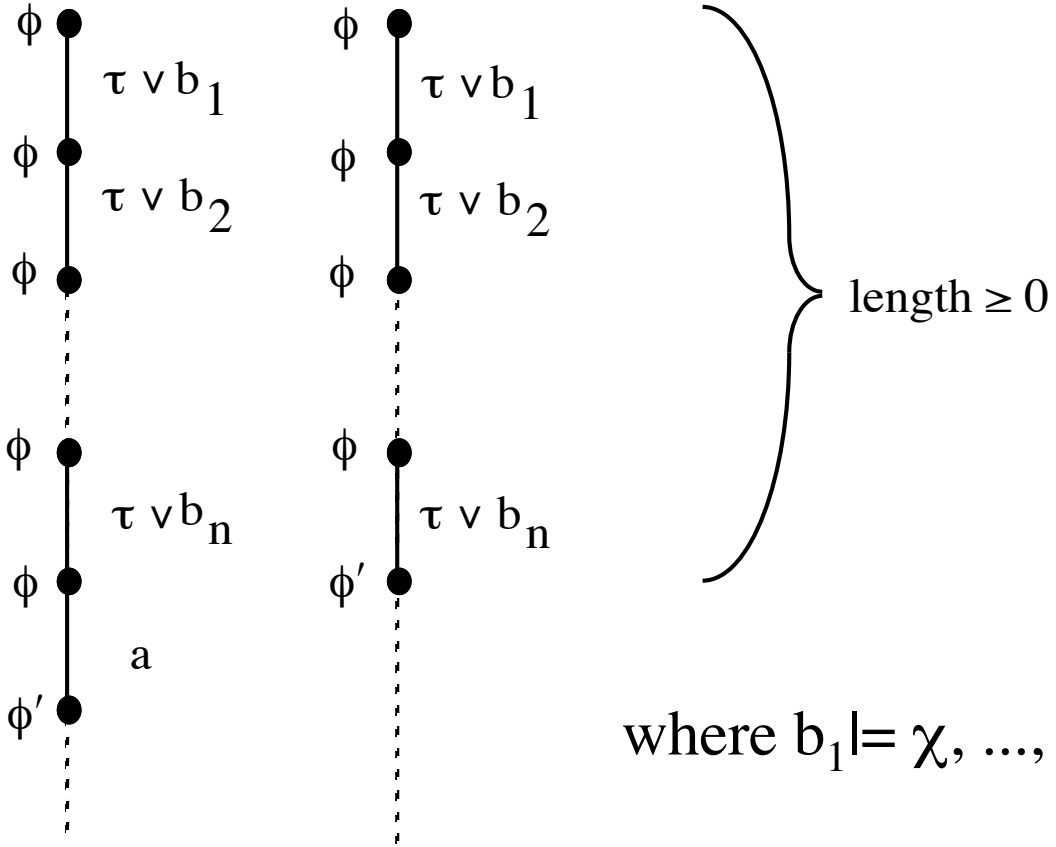
$\Pi(s)$ is the set of paths starting from s; a path is a sequence of successive transitions.

ACTL semantics

$s \models_{TS} true$		always;
$s \models_{TS} \phi \wedge \phi'$	iff	$s \models_{TS} \phi$ and $s \models_{TS} \phi'$;
$s \models_{TS} \phi \vee \phi'$	iff	$s \models_{TS} \phi$ or $s \models_{TS} \phi'$;
$s \models_{TS} \neg\phi$	iff	not $s \models_{TS} \phi$;
$s \models_{TS} E\gamma$	iff	there exists a path $\pi \in \Pi(s)$ such that $\pi \models_{TS} \gamma$;
$s \models_{TS} A\gamma$	iff	for all paths $\pi \in \Pi(s)$, $\pi \models_{TS} \gamma$;
$\pi \models_{TS} X_\chi \phi$	iff	$\lvert\pi\rvert \geq 1$ and $\pi(1) \in D_{\kappa(\chi)}(\pi(0))$ and $\pi(1) \models_{TS} \phi$;
$\pi \models_{TS} X_\tau \phi$	iff	$\lvert\pi\rvert \geq 1$ and $\pi(1) \in D_{\{\tau\}}(\pi(0))$ and $\pi(1) \models_{TS} \phi$;
$\pi \models_{TS} \phi\, {}_\chi U \phi'$	iff	there exists $i \geq 1$ such that $\pi(i) \models_{TS} \phi'$, and for all $1 \leq j \leq i-1$: $\pi(j) \models_{TS} \phi$ and $\pi(j+1) \in D_{\kappa(\chi)_\tau}(\pi(j))$;
$\pi \models_{TS} \phi\, {}_\chi U_{\chi'} \phi'$	iff	there exists $i \geq 2$ such that $\pi(i) \models_{TS} \phi'$ and $\pi(i) \in D_{\kappa(\chi')}(\pi(i-1))$, and for all $1 \leq j \leq i-1$: $\pi(j) \models_{TS} \phi$ and $\pi(j) \in D_{\kappa(\chi)_\tau}(\pi(j-1))$.

Action indexed Untils

$\varphi\ {}_{\chi}U_{\chi'}\ \varphi'$ $\varphi\ {}_{\chi}U\ \varphi'$



where $b_1 \models \chi, \ldots, b_n \models \chi$ and $a \models \chi'$

Derived modalities

Several useful modalities can be defined, starting from the basic ones.

$EF\phi$ for $E(tt_{tt} U \phi)$, and $AF\phi$ for $A(tt_{tt} U \phi)$;

these are called the *eventually* operators

$EG\phi$ for $\neg AF\neg\phi$, and $AG\phi$ for $\neg EF\neg\phi$;

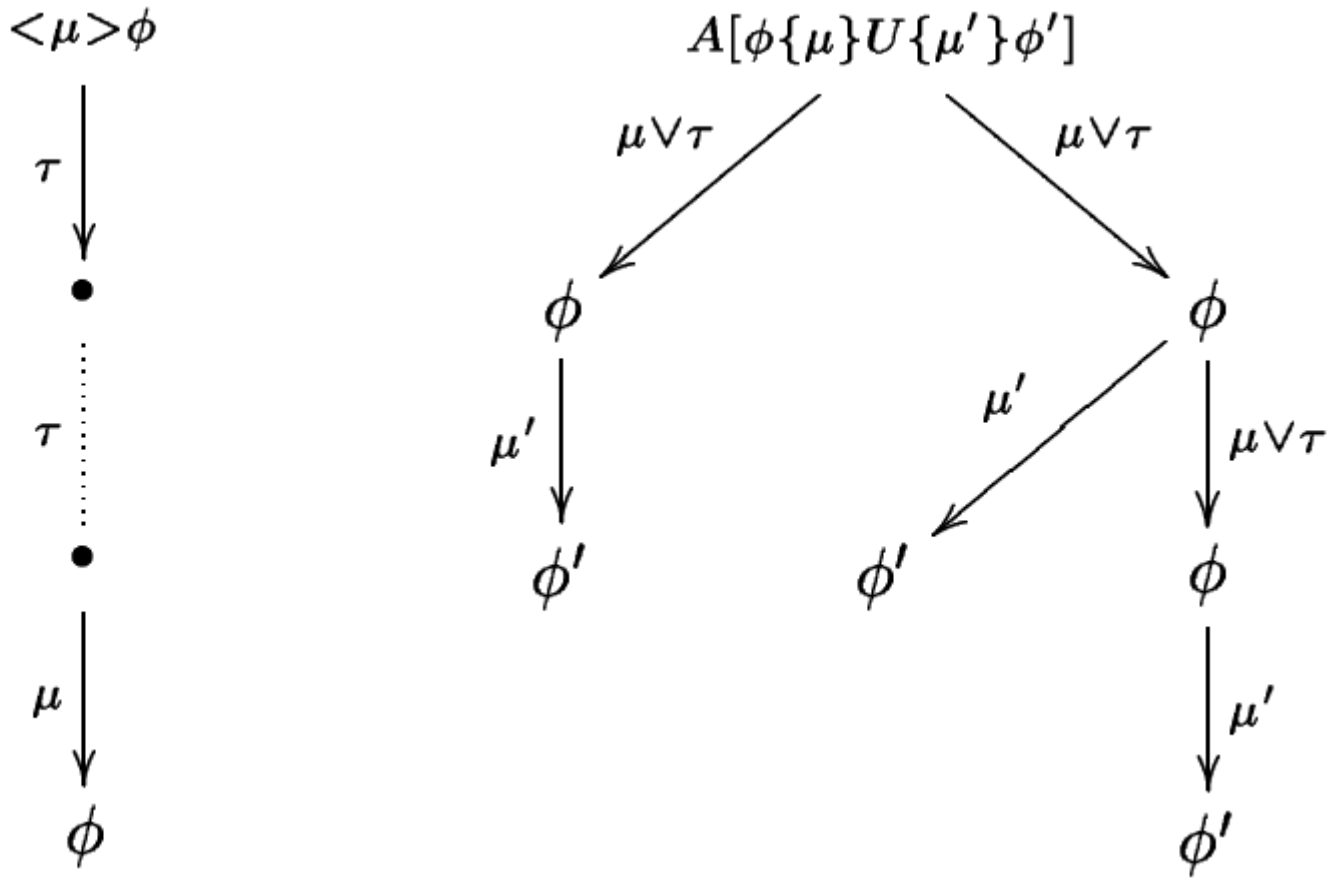
these are called the *always* operators

Hennessy-Milner "weak" modalities, [ ] < >:

<a>f = $E(tt_{ff} U_a f)$ **(There exists a visible transition labelled by a)**

[a]f = ¬ <a> ¬ f **(For all transitions labelled by a,….)**

Other examples



Safety and liveness properties

Classical distinction of properties of reactive systems:

Liveness properties (something good eventually happens)

Safety properties (nothing bad can happen)

$$EF\ EX_{good}\ true$$

$$AG\ \neg EX_{bad}\ true \quad (= \neg EF\ EX_{bad}\ true\ )$$

Safety and liveness properties

If a train leaves the level crossing (event leaving_t) then,

first, it has to approach to the level crossing (event approaching_t):

~ (EF <~ approaching_t> <leaving_t> true)

If a train approaches to the level crossing then it must immediately leave the level crossing:

AG ([approaching_t] AX {leaving_t} true)

If a car approaches the crossing (event approaching_c) then no train can approach the crossing until a car leaves (event leaving_c}) the crossing:

Alternatively a train cannot approach if before the car has not left:

AG ([approaching_c] A [true{~approaching_t} U{leaving_c} true])

Relations between ACTL and process algebras

• *A logic L is **adequate** with respect to an equivalence@ , if for every pair of processes q and q', q @ q' holds if and only if q and q' satisfy the same set of ACTL formulas.*

•The ACTL logic is adequate with respect to strong bisimulation equivalence on LTSs

• *Therefore minimization by strong bisimulation preserves ACTL formulae.*

•**Distinctive feature of ACTL and of Process algebras (vs. CTL/Kripke Structures); explicitation of the unobservable action Tau**

•*(in Kripke Structure internal moves are modelled by stuttering)*

•The ACTL-X (ACTL without next) logic is adequate with respect to branching bisimulation equivalence on LTSs

Basic model checking algorithm

- We present a model checking algorithm for (a subset of) ACTL, directly derived by the Clarke-Emerson-Sistla 1986 algorithm

- The algorithm proceeds visiting the automaton and labelling each state with the set of subformulae of the formula to be checked, that are satisifed in that state *(following the satisfaction relation |= used for defining the semantics of the logic).*

- The subset considered is composed of state formulae:

- P := true | ~P | EX{act}P | E[P{act}U P]

(Explicit) Model checking algorithm

```
for i =1 to length(p0)
 for each subformula p of p0  of length i
  case on the form of p
    p =  true    /* nothing to do */
    p  = q and  r:    for each s  in  S
       if q in L(s) and r in L(s) then add q  and  r to L(s)
       end
    p = ~q:              for each s in S
                           if q in L(s) then add  ~q to L(s)
                         end
    p = EX{a}q:          for each s in S
                         if (for some t in S, s-a→t, q in L(t)) then add EX{a}q toL(s)
                         end
    p = E[q {a}U r]:     for each s in S
                         if r in L(s)
                         then add E[q {a}U r] to L(s)
                         end
                         for j  = 1 to card(S)
                            for each s in S
                                    if  q in L(s) and (for some in S, s-a→t or s-τ→t, E[q {a}U r] in L(t))
                                    then add E[q {a}U r] to L(s)
                            end
                         end
    end of case
  end
end
```

Counterexample

- The *labeling* alghoritm permits, in the case of negative result, to find out the reason of the result, since it is possible to find all those states that do not verify some significant subformula, and hence contribute to the failure of the verification;

in general, model-checking alghoritms are able to provide a *counterexample*: for example, a path in the model that does not verify the (sub)formula.

A state and action semantic model for UCTL

Doubly Labeled Transition System: (Q, q0, Act*+{tau}, R, $\mathcal{L}$)

Where:

(Q, q0, Act*+{tau}, R) is a LTS

$\mathcal{L}$ is a labeling function $\mathcal{L}: Q \rightarrow \mathcal{AP}$

$\mathcal{AP}$ is a finite set of atomic propositions

(typically of the form VAR=value)

An action-state based logic: UCTL

- Action formulae:

$$\forall\chi ::= tt \mid event \mid \tau \mid \chi \wedge \chi \mid \neg \chi$$

- UCTL formulae:

$$\forall \qquad \phi ::= true \mid p \mid \neg \phi \mid \phi \wedge \phi \mid E\,\gamma \mid A\,\gamma$$

$$\gamma ::= X_{\chi}\,\phi \mid X_{\tau}\,\phi \mid \phi\ {}_{\chi}U_{\chi}\ \phi' \mid \phi\ {}_{\chi}U\ \phi'$$

UCTL semantics 1

transition label $\models$ Action formula

$\alpha \models tt$ holds always

$\alpha \models \neg \chi$ iff not $\alpha \models \neg \chi$

$\alpha \models \chi_1 \wedge \chi_2$ iff $\alpha \models \chi_1$ and $\alpha \models \chi_2$

$\alpha \models \tau$ iff $\alpha = tau$

$\alpha \models event$ iff $\alpha = \{e_1,\ldots,e_n\}$ $1 \leq n$ and

exists i, $1 \leq i \leq n$: $e_i = event$

UCTL semantics 2

state $\models$ formula

$q \models p$	iff $p \in \mathcal{L}(q)$
$q \models true$	holds always
$q \models \neg \Phi$	iff not $q \models \neg \Phi$
$q \models \Phi_1 \wedge \Phi_2$	iff $q \models \Phi_1$ and $q \models \Phi_2$
$q \models \mathbf{E}\gamma$	iff there exists a path π starting form q such that $\pi \models \gamma$
$q \models \mathbf{A}\gamma$	iff for all paths π_i, i=1…n starting form q $\pi_i \models \gamma$

UCTL semantics 3

path $\models$ formula

$\pi \models \mathbf{X}_{\{\chi\}}\ \phi$ iff $\exists$ q' such that first(π) $\xrightarrow{\alpha}$ q', and
q' $\models \phi$, $\alpha \models \chi$

$\pi \models \phi\ {}_{\chi}\mathrm{U}_{\chi'}\ \phi'$ iff $\exists\ q_1 \ldots q_n, \alpha_1 \ldots \alpha_n, , 0 \le n :\ q_n \models \phi'$
and for all i : $0 \le i < n$, $q_i \xrightarrow{\alpha_{i+1}} q_{i+1}$,
$q_i \models \phi$, $\alpha_i \models \chi$ and $q_{n-1} \xrightarrow{\alpha_n} q_n$
$\alpha_n \models \chi'$

$\pi \models \phi\ {}_{\chi}\mathrm{U}\ \phi'$ iff $\exists\ q_1 \ldots q_n, \alpha_1 \ldots \alpha_n, , 0 \le n : q_n \models \phi'$
and for all i : $0 \le i < n$, $q_i \xrightarrow{\alpha_{i+1}} q_{i+1}$,
$q_i \models \phi$ and $\alpha_i \models \chi$

UCTL semantics 3

γ Useful derived operators:

- $EF\ \phi \quad = E(true\ _{true}\ U\ \phi)$
- $EF_{\{\chi\}}\phi \quad = E(\ true\ _{\{\chi\}}\ U\ \phi)$
- $EF_{<\chi>}\phi \quad = E(\ true\ _{true}\ U\ _{\{\chi\}}\ \phi)$
- $AG\ \phi \quad = \neg\ EF\ \neg\ \phi$
- $AG_{\{\chi\}}\phi \quad = \neg\ EF\ _{\{\chi\}}\ \neg\ \phi$

- **UCTL is adequate w.r.t. strong bisimulation ≅**
- **on Doubly labeled transition systems**

- **UMC model checker exploits an on the fly algorithm**