MFES - Métodos Formais em Engenharia de Software

Model Checking

Ana Paiva

apaiva@fe.up.pt   www.fe.up.pt/~apaiva

Model Checking

Specification:

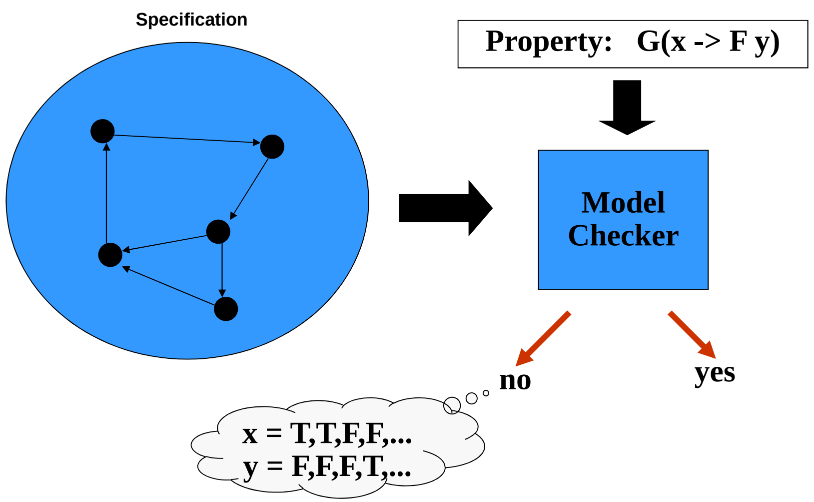
- The reality is described as a finite system of transitions
- The desired properties in temporal logic.

The aim is to assess whether the system is a model for the formula / requirement

System ⊨ requirement

by searching in all states.

Model Checking



Syntax of temporal logic

- Grammar of CTL*

$\phi, \psi ::= P1 \mid P2 \mid \ldots$ (Atomic propositions)

$\mid \neg\phi \mid \phi \wedge \psi \mid \phi \Rightarrow \psi \mid \ldots$ (Boolean operators)

$\mid X\,\phi \mid F\,\phi \mid G\,\phi \mid \phi\,U\,\psi \mid \ldots$ (Temporal operators)

$\mid E\,\phi \mid A\,\phi$ (Quantifiers)

Semantics of temporal logic (CTL*)

- Whereas an automaton
  - $A = \langle Q, T, q0, l \rangle$

    $T \subseteq Q \times Q$

    $l(q)$ = set of atomic propositions found in the state q
  - Satisfaction vs. unsuccessful

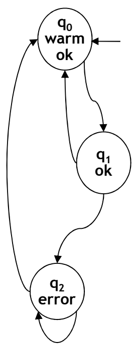
    $A,\sigma,i \models \phi$, at the moment *i* of the execution σ in A, ϕ is true (the context, A, is usually omitted)

    $\sigma,i \not\models \phi$, at the moment *i* if σ, ϕ is false
  - $\sigma(i)$ is the ith state of the execution σ

Semantics of temporal logic(CTL*)

$\sigma,i \models P$	iff $P \in l(\sigma(i))$
$\sigma,i \models \neg \phi$	iff it is not true $\sigma,i \models \phi$
$\sigma,i \models \phi \wedge \psi$	iff $\sigma,i \models \phi$ and $\sigma,i \models \psi$
$\sigma,i \models X\phi$	iff $i < \|\sigma\|$ and $\sigma,i+1 \models \phi$
$\sigma,i \models F\phi$	iff there is j such as $i \le j \le \|\sigma\|$ and $\sigma,j \models \phi$
$\sigma,i \models G\phi$	iff for all j's such that $i \le j \le \|\sigma\|$, and $\sigma,j \models \phi$
$\sigma,i \models \phi U \psi$	iff exists j, $i \le j \le \|\sigma\|$, such that $\sigma,j \models \psi$ and for all k's such that $i \le k < j$, and $\sigma,k \models \phi$
$\sigma,i \models E\phi$	iff exists σ' such that $\sigma(0)\ldots\sigma(i) = \sigma'(0)\ldots\sigma'(i)$ and $\sigma',i \models \phi$
$\sigma,i \models A\phi$	iff for all σ' such that $\sigma(0)\ldots\sigma(i) = \sigma'(0)\ldots\sigma'(i)$ and $\sigma',i \models \phi$

Example



Possible plays of the automaton:

σ_1: (q_0: warm, ok) $\rightarrow$(q_1: ok) $\rightarrow$(q_0: warm,ok)$\rightarrow$(q_1:ok) $\rightarrow$(q_0: warm,ok)…

σ_2: (q_0: warm, ok) $\rightarrow$(q_1: ok) $\rightarrow$(q_2: error)$\rightarrow$(q_0: warm,ok)$\rightarrow$(q_1:ok)…

σ_3: (q_0:warm,ok)$\rightarrow$(q_1:ok)$\rightarrow$(q_2:error)$\rightarrow$(q_2: error)$\rightarrow$(q_2:error)$\rightarrow$(q_2:error)…

σ_4: …

Proposition: mixture of atomic propositions and Boolean operators,
ex.: error $\Rightarrow \neg$ warm

Temporal operators: they allow to describe properties about sequences of states,

ex.: executions σ_1, σ_2 and σ_3 satisfy the property

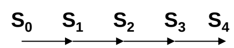
XXerror V XXXok ????

Languages for specifying properties

Linear Temporal Logic (LTL) – fragment of CTL* without the paths A and E

Model of linear time.

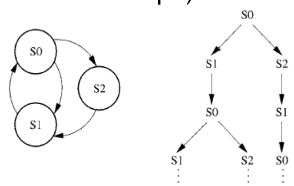
Temporal operators.



Computation Tree Logic (CTL) – fragment of CTL* in which the temporal operators (X, F, U, etc.) have to be directly related/connected (be under the immediate scope) with quantifiers A and/or E

Branching time model.

Operators more time quantifiers path.

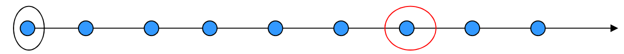


Timed CTL (TCTL)

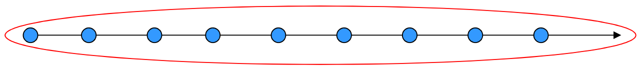
For real-time systems

LTL: temporal operators (future)

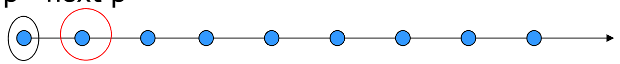
Fp or ◊p - finally p



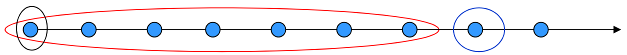
Gp or □p - globally p



Xp or ○p - next p



pUq - p until q



Formulas abbreviated

- GFp (allways there will be a state such that p)

$$\overset{\infty}{F}\, p$$

infinitely often

- FGp (all the time for a certain time onwards)

$$\overset{\infty}{G}\, p$$

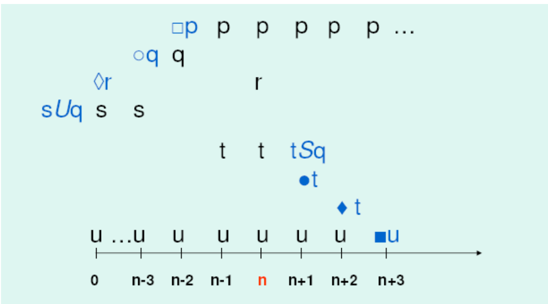
LTL: temporal operators (past)

♦p – Sometime in the past ($F^{-1}\phi$)

■p – Always in the past ($G^{-1}\phi$)

●p – In the previous state ($X^{-1}\phi$)

pSq – p since q



Properties

- ***Reachability*** property
  - states that some particular situation can be reached
- ***Safety*** property
  - expresses that, under certain conditions, something never occurs
- ***Liveness*** property
  - expresses that, under certain conditions, something will ultimately occur
- ***Fairness*** property
  - expresses that, under certain conditions, something will (or will not) occur infinitely often.
- ***Deadlock-freeness*** property
  - whatever the state reached may be, there will exist an immediate successor state.

Properties

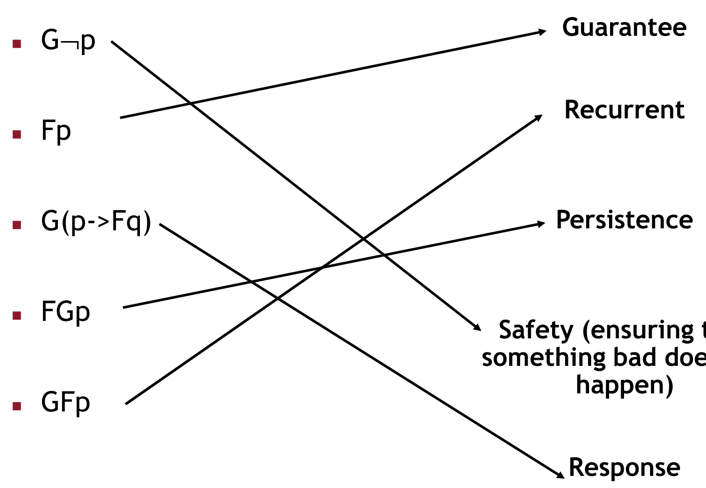
- Safety property
  - Example: A and B will never have simultaneous access to the critical region.

  A finite sequence can be used to prove its falsity
- Liveness property
  - Example: A and B will never have simultaneous access to the critical region.

  A finite sequence can be used to prove its falsity

  Example: If A wants to enter the critical region then it will happen.
  One can only prove the falsity by infinite sequences (since any finite sequence can be increased to satisfy the condition)
- Other examples:
  - The program ends?; The condition C1 is true until the condition C2 is established?;The conditions C1 and C2 are mutually exclusive?;The program P has no deadlocks?

LTL formulas

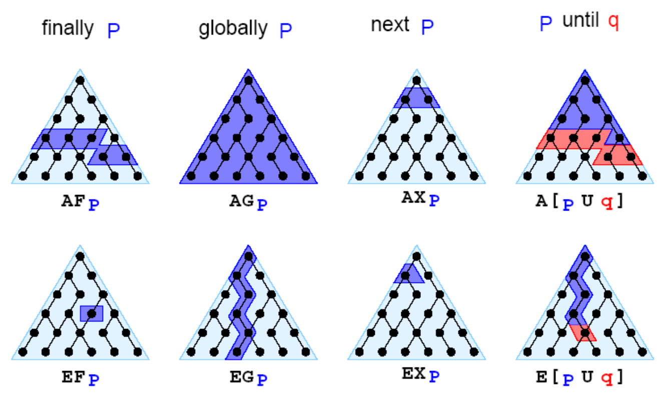


Until vs weak until

- x U y – Until: y holds at the current or a future position, and x has to hold until that position. At that position x does not have to hold any more.
- x W y - Weak until: x has to hold until y holds. The difference with U is that there is no guarantee that y will ever be verified. The W operator is sometimes called "unless".

Computation Tree Logic (CTL)

Temporal operators preceded by quantifiers.



AG p – Universal;

AF p – Inevitable;

EF p – Is possible;

AX p – the next state;

A[pUq) – p until q;

E – There is an execution;

A – for all executions

Expressiveness of CTL

- EX; AX; E_U_; A_U_; EF; EG; AF; AG; …
- Which of the CTL formulas are valid ?
  - Ep U A(P_2UP_3) valid
  - Ep U E(P_2UP_3) valid
  - Ep U (P_2UP_3) invalid
  - Ep U E(P_2UP_3) valid

Note:

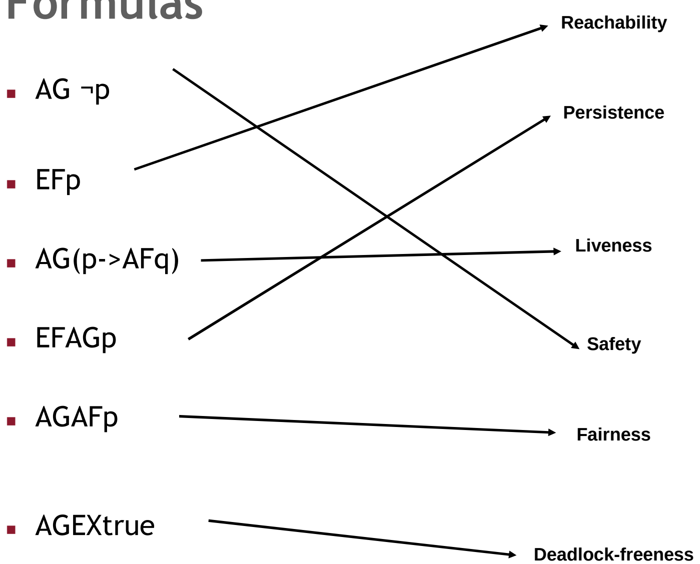
$\overset{\infty}{EF}\, p$ is not a formula in CTL because $\overset{\infty}{F}$

is an abreviation of GF.

De Morgan CTL Laws

- De Morgan's law
  - ¬ AFϕ ≡ EG ¬ ϕ
  - ¬ EFϕ ≡ AG ¬ ϕ
  - ¬ AXϕ ≡ EX ¬ ϕ
- Quantifiers F and G can be written by use of the operator until (U)
  - AFg A(true U g)
  - EFg E(true U g)
  - AGg ¬ E(true U ¬ g)
  - EGg ¬ A(true U ¬ g)

Formulas



Properties

- Examples:
- Safety condition (something bad does not happen)
  - There may be two green lights lit in two different streets at the same time
  - AG~(G1 /\ G2)
- Fairness Condition (something good does happen)
  - In the future, one of the streets will be lit green
  - EF (G1 \/ G2)
- "Weak until" whatever the behavior (A), the car never start walking (starts) as (W) the key is not placed in the ignition (key)
  - A¬starts W key = (¬starts U key) \/ G¬starts

Quantifiers

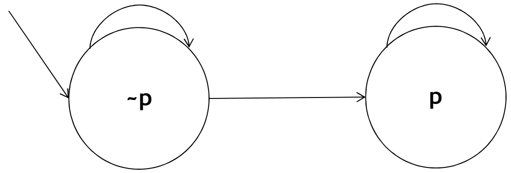
A: Throughout all executions

E: During an execution

What is the difference between AGFp and AGEFp?

- AGFp = $\overset{\infty}{AF}\ p$: over all executions (A), in every moment of time (G), we will find later (F) a state satisfying p. Thus, p must be met repeatedly (infinitely often).
- AGEFp: In any moment of any execution should be possible to come to satisfy p, i.e., p is always potentially achievable, even if an execution in which p is ever made. Throughout all the runs, the second quantifier, E, to express the fact that there are alternatives that allow plays to get different behaviors of the system (e.g., where there is p)

Quantifiers AGFp vs AGEFp



This CTL state machine satisfies AGEFp but not AGFp

Properties – summary

- Reachability (*"atingível"*) EFϕ
  - A given state / situation is attainable
- Safety (*"segurança"*) AG$\neg\phi$
  - Under certain circumstances, a situation never occurs
- Liveness (*"certeza, resposta"*) AG(req => AFsat), AGEF*init*
  - Under certain circumstances, a given situation will occur
- Fairness (*"justiça, recorrência"*) AGAFϕ
  - Under certain circumstances, a given situation will (or not) occur repeatedly (infinitely often)
- Deadlock-freeness AGEX*true*
  - For any state, there is always a successor state

Timed Temporal Logic (TCTL)

- Formal grammar of TCTL

$\phi, \psi ::= P1 \mid P2 \mid \ldots$ (atomic propositions)

$\mid \neg\phi \mid \phi \wedge \psi \mid \phi \Rightarrow \psi \mid \ldots$ (Boolean operators)

$\mid EF_{(\sim k)}\phi \mid EG_{(\sim k)}\phi \mid E\phi\ U_{(\sim k)}\psi$ (time operators)

$\mid AF_{(\sim k)}\phi \mid AG_{(\sim k)}\phi \mid A\phi U_{(\sim k)}\psi$

~ a symbol of comparison $\{<, \leq, =, >, \geq\}$ and k any rational number $\mathbb{Q}$

$PU_{(<2)}Q$ means that P is true until Q and that Q is true in two units of time from the present moment

Model Checker

- To implement a Model Checker:
  - Build an atomaton S: AS
  - Build the automaton of the negation of property ~P: A~P
  - Calculate AS ∩ A~P
    - If Ø then P is true.
    - If ≠ Ø then the obtained sequence of transitions is a counter-example of P.

Example (model checker)

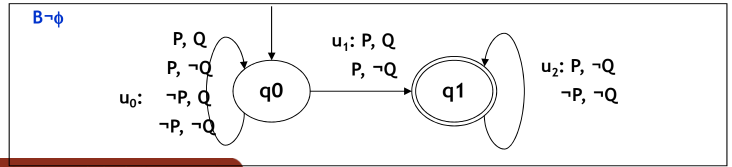
For a given formula ϕ the model checker builds an automaton $B\neg\phi$ that recognizes executions that does not satisfy ϕ.

$A \otimes B\neg\phi$ is the automaton with executions of A that does not satisfy ϕ.

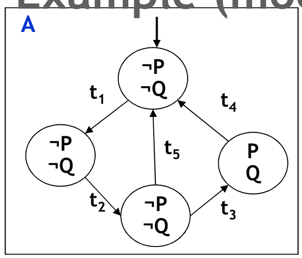
Example:

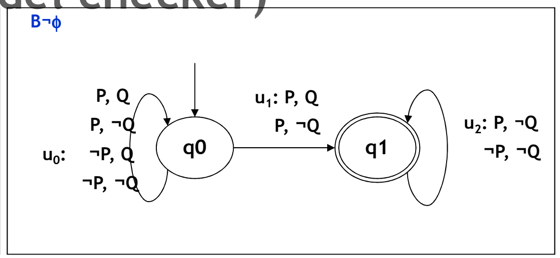
$\phi = G(P \Rightarrow XFQ)$: Any occurrence of P must be followed (later) by an occurrence of Q

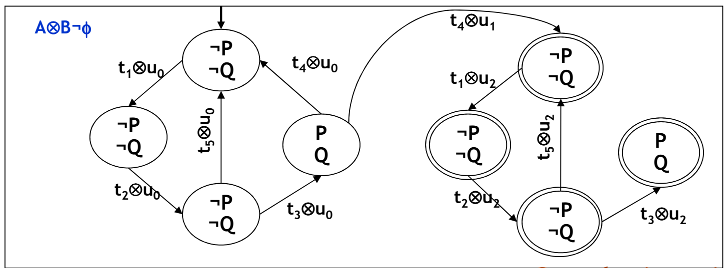
$\neg\phi$ means that there is an occurrence of P after which no more is Q.



Example (model checker)







Conclusion: A ⊭ ϕ

Tools

SPIN

http://netlib.bell-labs.com/netlib/spin/whatisspin.html

SMV (Symbolic Model Verifier)

http://www.cs.cmu.edu/~modelcheck/smv.html

HYTECH (Linear Hybrid Systems)

http://www.eecs.berkeley.edu/~tah/HyTech

UPAAL (Real-Time Systems)

http://www.upaal.com)

Kronos (Real-Time Systems)

http://www-verimag.imag.fr/TEMPORISE/kronos