Temporal and Functional Analysis

UNS International Master1 Lecture 4

Frédéric Mallet

Robert de Simone

temporal logics

Temporal logics

- Classical logic + modalities (future mostly)
- Properties true « in certain states » or « for certain runs/ paths »
- **linear** or *tree-like* **branching** time vision
- Property classes (not exhaustive but very frequent):
  - *safety*/sûreté: « *nothing bad happens* »
  - *liveness*/inevitabilité:« *eventually something good occurs*» (after a finite, but unknown/unbounded period); shows progress out of livelock loops.
  - *fairness*/équité ? *liveness assumption*
    - *Weak: provided almost always P*
    - *Strong: provided unfinitely always P*

Modalities: width: E: possibly (in one future)
A: absolutely (for all future)
depth X: next
F: eventually
G: forever
U: until *(binary op)*

Examples

- No overflow (safety)

  $AG(x < maxint)$

- Eventually I will have to start while not ready (reachability)

  $EF(Start? \land \neg Ready)$

- A request will always be acknowledged afterwards

  $AG(Req \Rightarrow AF\ Ack)$

- Each request can be acknowledged afterwards

  $AG(Req \Rightarrow EF\ Ack)$

- A data written is read before the next write (no lost)

  $AG\ (Write \Rightarrow (\neg Overwrite\ U\ Read))$

- Infinitely often $AG(AF\ Event)$
- System is resettable («attractivity »)

  $AG(EF\ restart\_state)$

CTL* : syntax

- State and Path formulae (with corresponding interpretation)
  - State formulae (f,g, …) :
    - atomic predicates (abstraction of data or control predicates)
    - $\neg f$ (not f), $f \wedge g$ (f and g), $f \vee g$ (f or g)
    - E(xists) p, A(ll) p *(width),* with p a path formula
  - Path formulae (p,q, …) :
    - include state formulae f,g,…
    - $\neg p$, $p \wedge q$, $p \vee q$
    - X p (next), p U q (until)
    - F p (eventually), G p (always) *(depth)*

CTL* : semantics

M,s |= basicpred	iff	basicpred labels s
M,s |= ¬ f	iff	not (M,s |= f)
M,s |= f∧g	iff	M,s |= f and M,s |= g
M,s |= E p	iff	∃ π path from s, M,π |= p
M,s |= A p	iff	∀ π path from s, M,π |= p
M,π |= f	iff	M,s |= f , where s initial state of π
M,π |= X p	iff	$M,\pi\|_1$ |= p (π$\|_1$ is π stripped of its first step)
M,π |= F p	iff	∃ *i*, M,π$\|_i$ |= p (π$\|_i$ is π without i first steps)
M,π |= G p	iff	M,π |= ¬ F ¬ p
M,π |= p U q	iff	∃ *i*, M,π$\|_i$ |= q, et ∀*j*< *i* , M,π$\|_j$ |= p

Linear-time vs branching-time temporal logics

- CTL (computation-tree logics)
  - Only state-formulae: AG, AF, AX, A(fUg),
    EG, EF, EX, E(fUg)
- LTL (linear-time logics)
  - Only path formulae (with: basicpred valued on path as at initial state)
  - then M |= p iff ∀ π path in M, M,π |= p

- CTL : polynomal complexity, direct application *on model, by state state/predicate transformation*
- LTL : exponential on the formula size only, observers *as Büchi automata*

Expressivity: CTL vs LTL

- No way in LTL to speak of branching stages

$$(EX\ Q \wedge EX\ \neg Q),$$

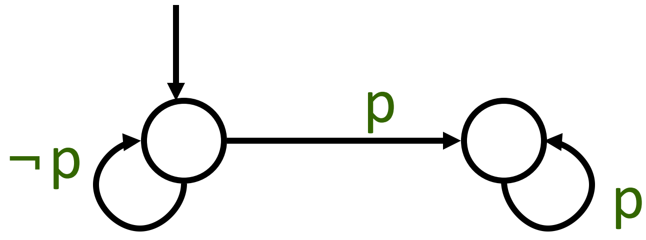
(possibly next Q and possibly next not Q)

- No way in CTL to impose that various subformulas all deal with « a single » future path

$$GF\ p$$

(infinitely often p),

while the «CTL version» $AG(EF\ p)$ is satisfied by:



Model-checking of temporal properties

(on finite models)

Model-checking

- Check formulae on given models !

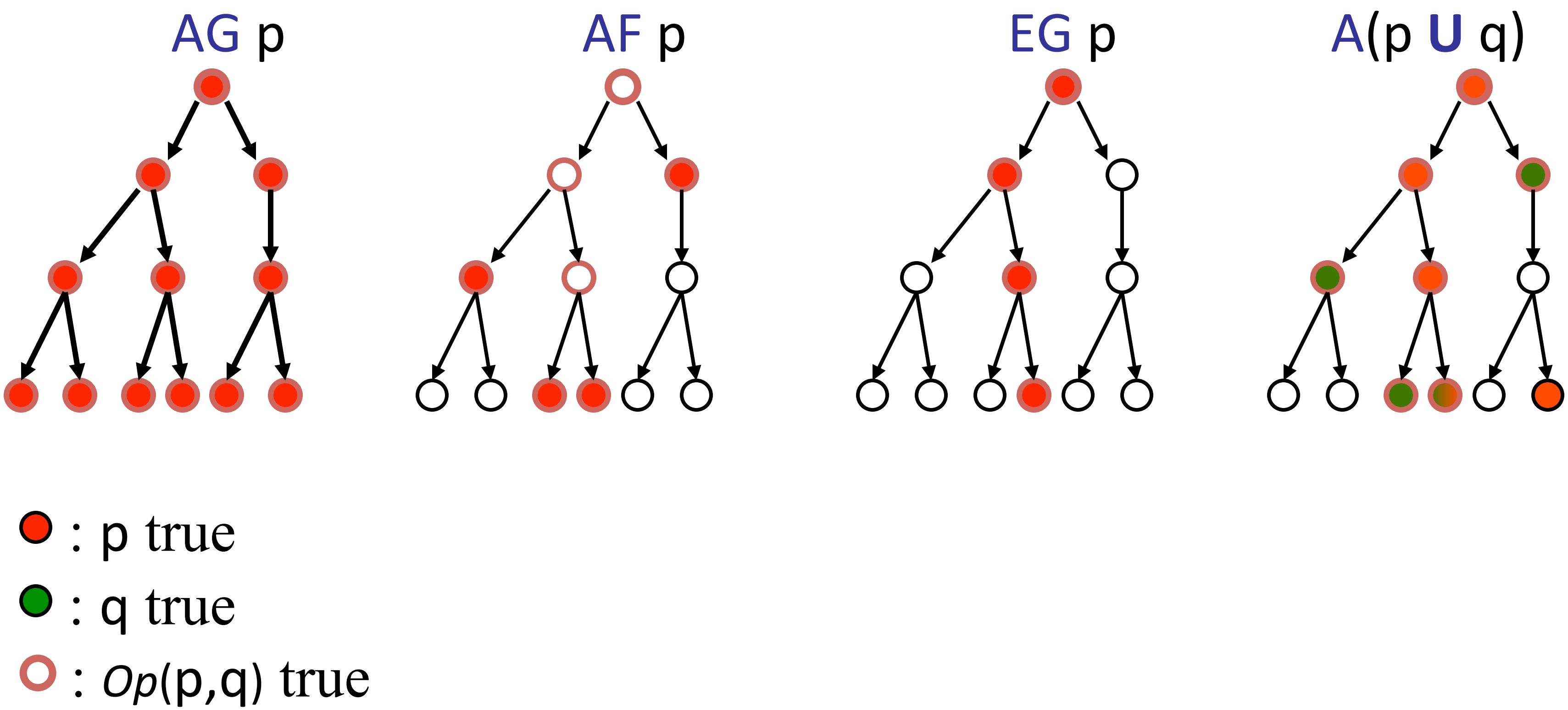
(as opposed to: verify whether the formula can accept a model (satisfiability)

  – Our models are finite state machines

  – issues are:
    - generation and search of reachable states and runs)
    - ➢Fixpoint algorithms finite state → convergence (Tarski th.))

Computation Tree Logic

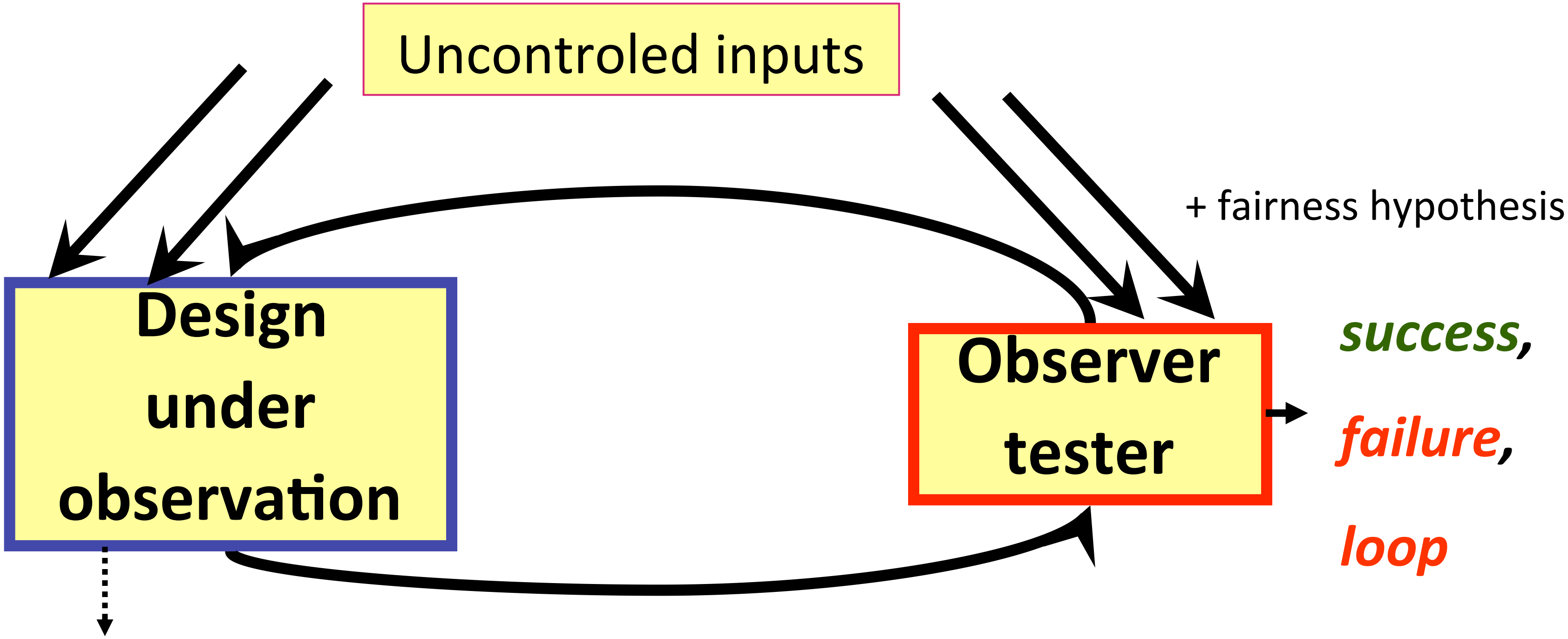
- Intuitive algorithm presentation (sketch)
- modalities: AX p , EX p, A(p **U** q) , E(p **U** q), ...



CTL model-checking: formal algorithm definition

- Smallest or largest fixpoints:
  - $AF\ p = \mu Z.\ p \vee AX\ Z$
  - $EF\ p = \mu Z.\ p \vee EX\ Z$
  - $AG\ p = \nu Z.\ p \wedge AX\ Z$
  - $EG\ p = \nu Z.\ p \wedge EX\ Z$
  - $A[p\ U\ q] = \mu Z.\ q \vee (p \wedge AX\ Z)$
  - $E[p\ U\ q] = \mu Z.\ q \vee (p \wedge EX\ Z)$

LTL Observers



Brings down

safety to (un)reachability, and

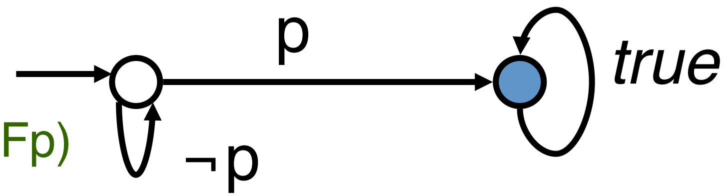
liveness to existence (or not) of «non-terminating» fair loops

in the composed system

LTL model-checking

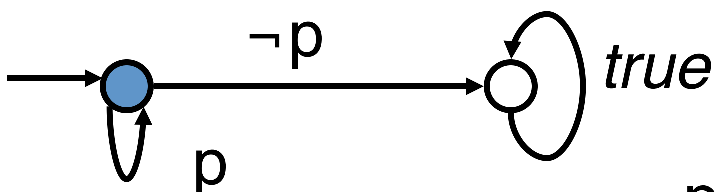
- One unfold definitions to create new states (states are named after residual subformulae)
- A run is successful if it crosses infinitely often a success state (painted blue)

  - F p
  $= p \vee (\neg p \wedge XFp)$



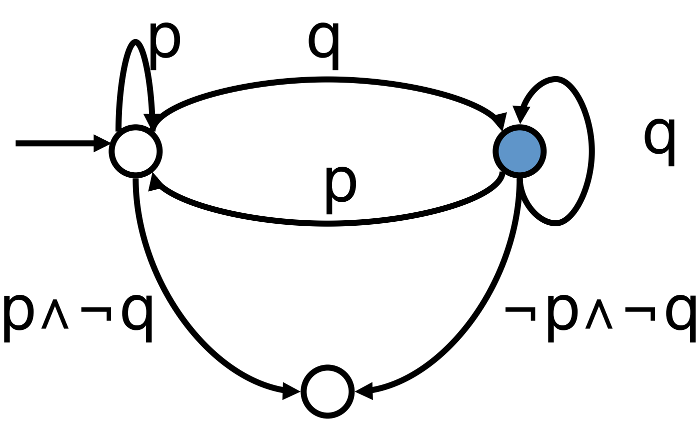
Success (blue state)

  - G p
  $= p \wedge XGp$

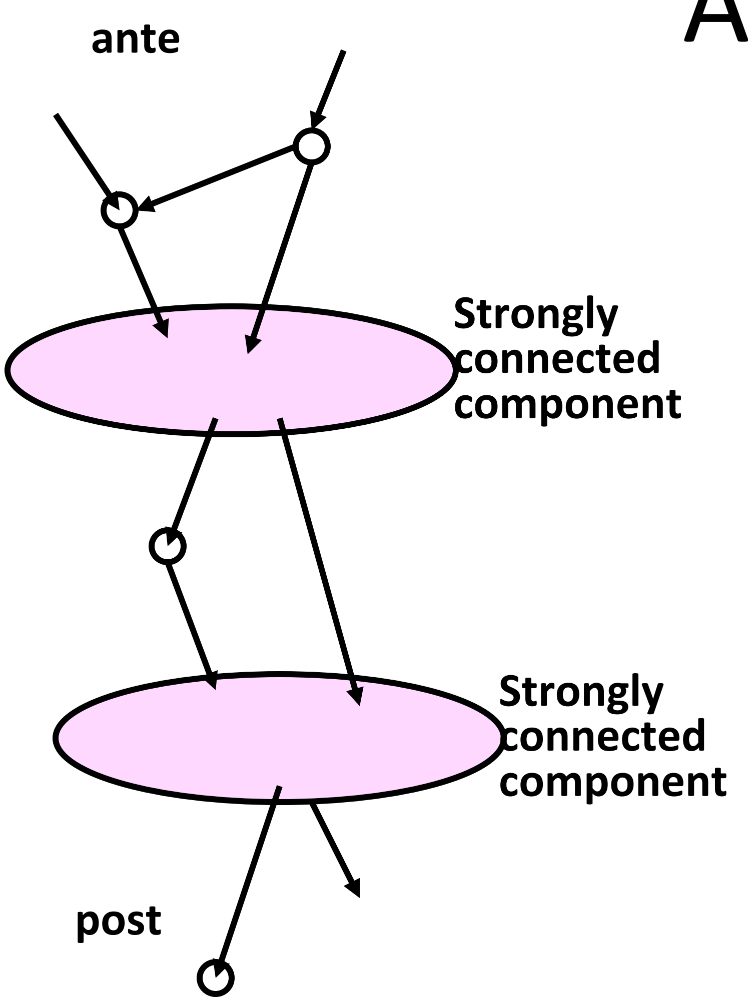


Failure (not-blue state)

  - G (p U q)



Approximating loops



First (smallest) fixpoint

(remember X is neXt state)***:***

$\mu Z.$ Init $\cup$ X(Z)

provides the reachable « playground » zone R

Second (largest) fixpoint:

$\nu Y.$ R $\cap$ X(Y)

computes SCCs, with their outgoing states

But this is empty iff SCCs are !

Symbolic state space representation

Reachable state space construction

- Global state = local states vector
- Concurrency: combinatorial explosion
- In principle, exhaustive depth-first or breadth-first search (with visited states recollection)
- Optimizations
  - Symbolic state space representation (SMV)
  - Compositional methods (SMV)
  - Conservative approximations
  - On-the-fly and partial order techniques (SPIN)
  - Partitioned transitions:
    - asynchronous processes : local actions
    - synchronous processus : local registers (SMV)

Binary Decision Diagrams

- Discrete types (boolean, bounded integers → *bitsets (encoding states, transitions)*
- Sets of …

  → bitset predicates on boolean variables → boolean formulae

  → *BDDs*
- Canonical graphs (unique normal form)

Generalized XOR « x ⊕ y ⊕ z »