Static Analysis of Programs
Model Checking
MSI Option

Olivier Hermant
olivier.hermant@mines-paristech.fr

MINES ParisTech, Centre de Recherche en Informatique

September 2020

Overview

Classical Programs

- compute
- terminate
- return a result
- complex data, but sequential flow

Reactive Programs

- *must* not terminate
- does not return anything
- simple data, distributed workflow

Examples

Operating Systems, control-command systems (CAS), $\cdots$

Classical Program

- complex predicates, but fixed temporal aspect
- "the array is sorted"

Reactive Program

- simple predicates, but various temporal aspects
- "if a process continuously asks for execution, the OS will execute it"
- "it is always possible to get back to the initial state"
- "every time a failure is detected, an alarm is emitted"
- "every time an alarm is emitted, a failure has been detected"

Model Checking

Automated verification technique for reactive systems

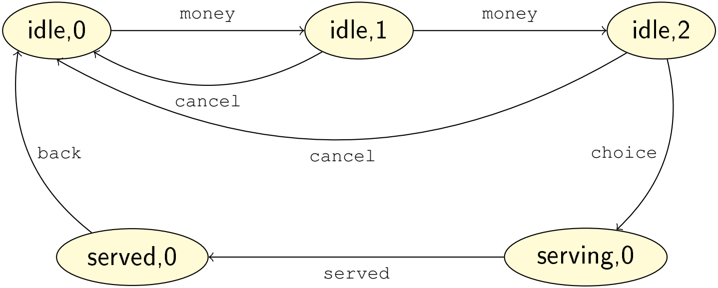
Principle

1. modelize the system with M and the property by φ
2. does $M \models \varphi$?
3. Analyze
   - yes, ok. But beware that we considered only a model !
   - no, we have a counterexample. Replay it on the real system
     - real bug
     - otherwise, refine M along φ

- applicable during verification and design phases
- automatized
- lesser costs
- geared towards bug finding, more exhaustive than tests

- 1975 : verification is not adapted to reactive systems
- 1977 : Pnueli proposes to use temporal logics
- 1981 : CTL Model Checking by Clarke, Sifakis, ...
- '80 & '90 : theoretical results
- '90 & '00 : performance improvement and extensions (probabilities, time)
- '00 : wide adoption and standardisation
  - microchip design (Intel, ...)
  - standardisation of the language PSL
  - *software* Model Checking (MS)
- '07 : Turing Prize to Clarke, Sifakis, Emerson

- transition system : modelizes the behavior of a reactive system



- **Accesibility**. A given situation can be reached.
  x can have the value 0, every instruction can be executed
- **Invariance**. Each state respect a good property.
  x is never 0, there is no array overflow
- **Safety**. Something bad never happens.
  I can get money only if if have the right PIN
- **Liveness**. Something good eventually happens.
  the program terminates, the message is eventually transmitted, the program always come back to the initial state
- **Fairness**. Something good repeats infinitely often.
  if a process asks to be executed indefinitely, it will be executed infinitely many times all philosoph eats infinitely many times
- **Observational Equivalence**. Do two systems have the same behavior ?
  simple system vs. optimized system

How to Represent Properties

- use temporal logics !
- advantages :
  - natural language imprecise, ambiguous
  - generic
  - automated verification possible
- several possible temporal logics :
  - LTL
  - CTL
  - CTL*
  - ...

Shortcomings of Model Checking

1. **Finite** transition systems
   - sometime hard to get a model
   - finite domain for variables, finite number of tasks, ...
2. **Small** transition systems
   - algorithmical and modelization problem
   - example : 10 variables on 8 bits : 10^{256} possibilities
     - one possibility is 10 bytes
     - all possibilities is 10^{245} TB

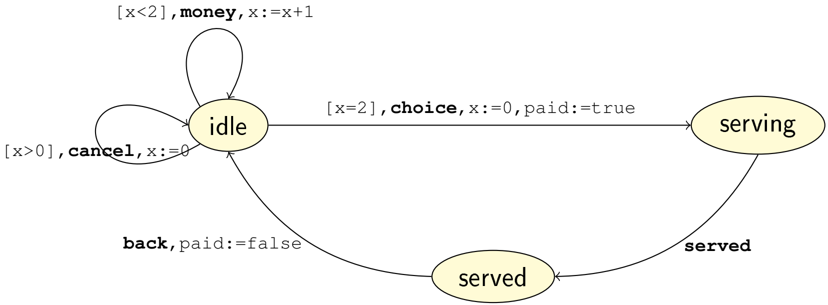
- Model Checking must manage state explosion

- finite systems $\Rightarrow$ bounded variables
- typically :
  - communication protocol
    - difficulty : intertwining of the agents
  - microprocessors
    - difficulty : number of variables
- in the future :
  - web services
  - small software, like drivers

- cost of bugs : \$64 billion/y in the US
  - Toyota acceleration bug : \$1.2 billion
- Cost of verification
  - \$10 billion/y for tests in the US
  - 50 kLOC to debug : 60 days, \$30,000 and indirect costs
  - more that 50% of the cost of a critical system
  - for standard software, 30% in average
- Cost of removing a bug
  - 1 at design-time
  - 5-15 at unit testing time
  - 15-90 at integration time
  - 50-200 at production time
- Bug repartition (automotive industry, Bosch)
  - 60% specification, 20% design, 10% code, 10% system

Modelization

- abstraction of a reactive system

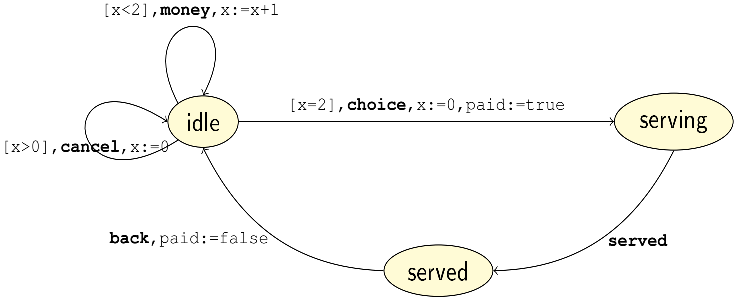


- Control states : `idle`,`serving`,`served`
- Variables : `x:int`, `paid:bool`
- Transitions : `money`, `choice`,`served`,`back`,`cancel`

State Machine

$P = \langle C, V, A, T \rangle$

- C finite set of control states
- V finite set of variables
- A finite set of `[guarded]` **`labelled`** `actions` on V
- $T \subseteq C \times A \times C$ finite set of transitions



Transition System

$S = \langle Q, T, \to \rangle$

- Q : set of states, or configurations
- T : set of labelled transitions
- $\to \subseteq Q \times T \times Q$ transition relation

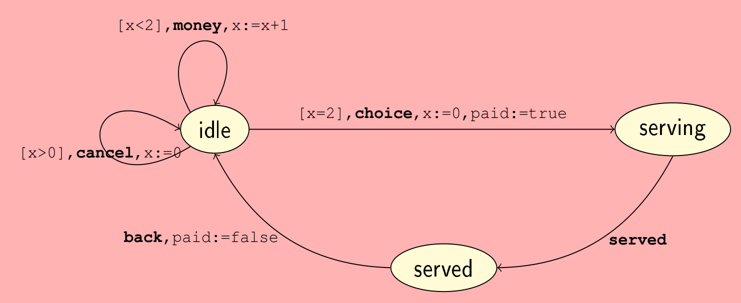
- Q represents the set of possible states of the system
- a transition t transforms a state q in a state q' if $(q, t, q') \in \to$

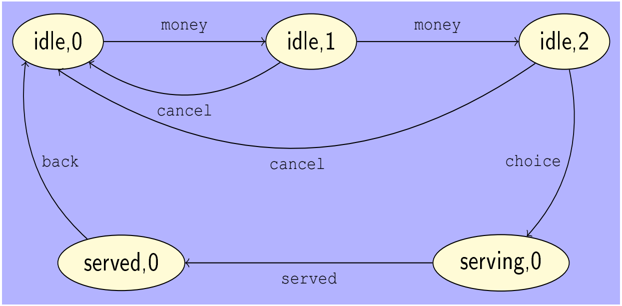
From State Machines towards Transition Systems

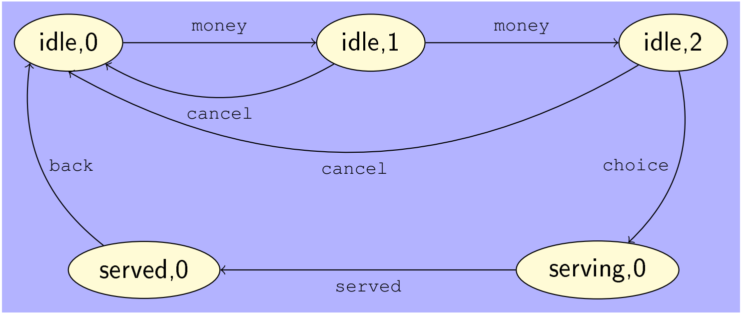
- Equivalence between a state machine $P = \langle C, V, A, T\rangle$ and a transition system $S = \langle Q, T, \rightarrow\rangle$
- a transition system has no variable (no state) : incorporate it inside Q !
- From P to S :
  - assign to variables $v_i \in V$ a domain D_i

$$D = D_1 \times \cdots \times D_n$$

  - associate actions $a \in A$ with partial functions $[\![a]\!] : D \rightarrow D$
    - **partial** because of the guard conditions
  - set $Q = C \times D$, a configuration can be written as (c, d)
  - let $((c, d), t, (c', d')) \in \rightarrow$ iff $t = (c, a, c')$ **and** $[\![a]\!](d) = d'$.


[x<2],**money**,x:=x+1
[x=2],**choice**,x:=0,paid:=true
idle
serving
[x>0],**cancel**,x:=0
back,paid:=false
served
served


idle,0
money
idle,1
money
idle,2
cancel
back
cancel
choice
served,0
serving,0
served



- $(i,0) \xrightarrow{\texttt{money}} (i,1) \xrightarrow{\texttt{money}} \cdots$
- can be shortened in `money`, `money`, `choice`, $\cdots$
- $\mathcal{L}(S)$ = Language of S = set of executions of S

- Reactive system = real world system
- State machine P = model syntax
- Transition system S = semantics of P

Temporal Logics

Temporal Properties

- **Accessibility**.
  A given situation can be reached
- **Invariance**.
  Each local state respects a good property
- **Safety**.
  A bad thing never happens
- **Liveness**.
  Something good eventually happend
- **Fairness**.
  Something good repeats infinitely often
- **Observational Equivalence**.
  Two systems are equivalent

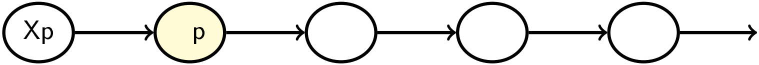
Express Properties

- need to express families of properties, not only particular cases
- Natural languages
  - imprecise
  - **and** verbose
- Graphical Formalisms
  - more precise
  - concise
  - easy to learn and share
  - still lack of precision and expressiveness

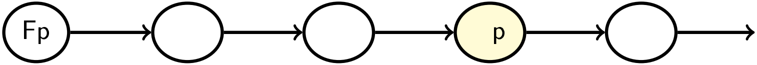
- Why logics ?
  - no ambiguity in the expected properties
  - allow to prove soundness of the system
- Why **temporal** logics
  - concision, expressiveness, simplicity
  - algoritmics : decision and complexity
- **What** temporal logics ?
  - classical logic
  - propositional, no first-order
  - temporal connectives **and** path quantification

Express a succession of events along an (execution) path :

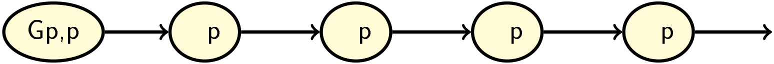
- X – ne**X**t : "at the next step"



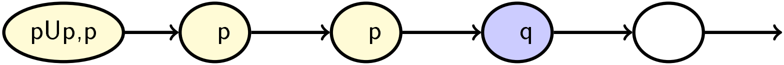
- F – **F**inally : "in some moment in the future"



- G – **G**lobally : "at any moment in the future"



- U – **U**ntil : "until"



- **Accessibility**.
  - A certain situation can be reached
  - $F(x = 0)$
- **Invariance**.
  - Each local state respects a good property
  - $G\ \neg(x = 0)$
- **Liveness**.
  - Something good eventually happens
  - $G(p \Rightarrow Fq)$
- **Total Soundness**.
  - $(\text{init} \wedge \text{precondition}) \Rightarrow F(\text{end} \wedge \text{postcondition})$

Path Quantifiers

- Temporal Connectives
  - consider one execution at a time
  - executions are independent from each other
  - they are organized in a set
  - future is determined
- One may want to speak about possible futures, along the actions taken by the system
  - some states have the choice between different futures
  - interdependents executions
  - organized in a tree
- We introduce path quantifiers
  - **A** : all the future paths
  - **E** : at least one future path

Path Quantifiers and Temporal Connectives

- **EFp** : p is true in at least one future state
- **AFp** : p is reachable on every path
- **EGp** : there exists a path with p always true
- **AGp** : p is always true

- **Accessibility**.
  - A certain situation can be reached
  - $EF(x = 0)$
- **Invariance**.
  - Each local state respects a good property
  - $AG\ \neg(x = 0)$
- **Liveness**.
  - Something good eventually happens
  - $AG(p \Rightarrow Fq)$
- **Total Soundness**.
  - $A((\text{init} \wedge \text{precondition}) \Rightarrow F(\text{end} \wedge \text{postcondition}))$
- **Fairness**.
  - $A(GF(\text{execution request}) \Rightarrow GF(\text{process scheduled}))$

Several Temporal Logics

- Linear vs. Branching
  - restricts the ability to speak about possible futures
- Expressiveness
  - Syntactic or Semantic
- Conciseness
- With or without past
- Verification complexity
- Less formalized factors
  - understanding, adequation to needs

- the composition of a logic $\mathcal{L}$:
  - a set L of formulas (φ)
  - a domain D for interpretations (noted I)
  - a satisfiability relation $\models \subseteq D \times L$
- We say that :
  - I is a model of φ iff $I \models \varphi$
  - φ is satisfiable iff there exists I s.t. $I \models \varphi$
  - φ is valid if for all I, I satisfies it
  - φ is contradictory iff no I satisfies it
- let $\mathcal{L}$ be a logic and $\varphi \in L$
  - $[\![\varphi]\!]$ is the set of interpretations satisfying φ
  - $E \subseteq D$ is $\mathcal{L}$-definable iff there exists $\varphi \in \mathcal{L}$ s.t. $E = [\![\varphi]\!]$.

Finally, we got to Semantics !

- $I \models \varphi$ is typically a semantic relation
- So, no syntactic proof system
- actually, proof system is not relevant here
- and dealing with semantics is **OK** in this case :
  - no first-order quantification **OK**
  - verify whether φ is satisfied in **one** particular model M. **OK**

- on a formula φ :
  - **Model Checking** : do we have $I \models \varphi$?
  - **Satisfiability** : is φ satisfiable ?
  - **Validity** : is φ valid ?
  - **Synthesis** : Find I such that $I \models \varphi$.
- on logics $\mathcal{L}_1$ and $\mathcal{L}_2$:
  - **Expressiveness** : do $\mathcal{L}_1$ and $\mathcal{L}_2$ define the same sets ?
  - **Conciseness** : do $\mathcal{L}_1$ and $\mathcal{L}_2$ define identical sets with formulas of comparable size ?

- you already know the syntax
- interpretation domain : boolean valuations

$$\text{for each } A_i, I(A_i) = 0 \text{ or } 1$$

- Satisfiability relation :
  - $I \models \top$
  - $I \not\models \bot$
  - $I \models A_i$ iff $I(A) = 1$ (A atomic)
  - $I \models \varphi_1 \wedge \varphi_2$ iff $I \models \varphi_1$ and $I \models \varphi_2$
  - $I \models \varphi_1 \vee \varphi_2$ iff $I \models \varphi_1$ or $I \models \varphi_2$
  - $I \models \neg\varphi$ iff $I \not\models \varphi$
- Examples : $A \vee \neg A$ is valid (excluded-middle !), A is satisfiable, $A \wedge \neg A$ is contradictory

- we have a syntax for propositional logic
- we have a proof system for propositional logic
- we have a semantics for propositional logic
- we can ask ourselves about soundness and completeness !

Soundness

If $A_1, \cdots, A_n \vdash B$ and $I \models A_i$ then $I \models B$

- **YES**, it holds.
- proof ? Exercise ...

Completeness

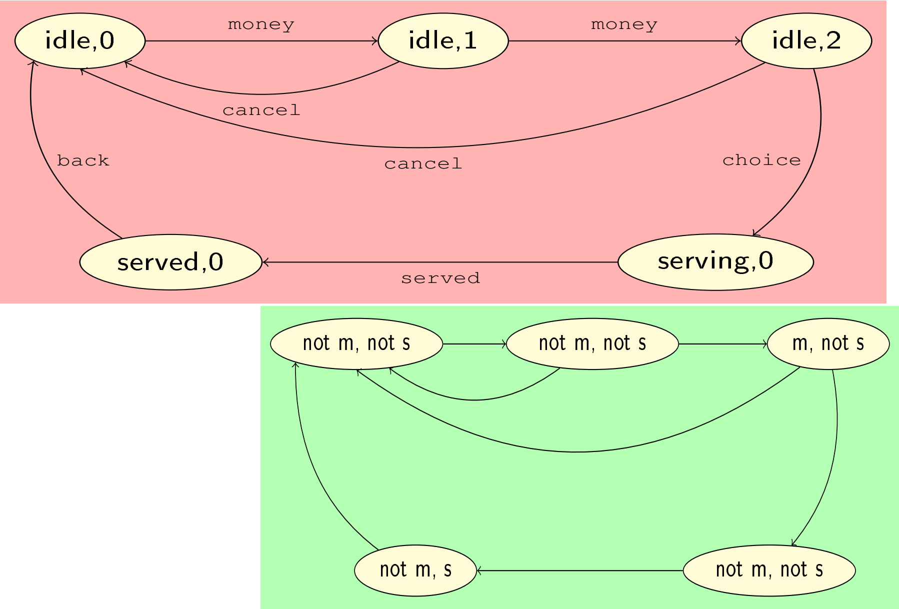
If , for any I, $I \models A_i$ then $I \models B$, then $A_1, \cdots, A_n \vdash B$

- **NO**, it does not hold.
- we cannot build a proof of the excluded-middle : $\vdash A \vee \neg A$
- add this principle to the logic, and
- **YES**, it holds.

- Main domain of interpretation
  - Kripke structures M
  - for us = a transition system S
- We define the satisfiability relation $\models$ in three steps, the first two of which are interdependent :
  1. satisfiability of a path formula (no head **A** or **E**) on a path
  2. satisfiability of a state formula (with a head **A** or **E**) on (M, s) based on the previous point
  3. M satisfies φ if (M, s^0) satisfies φ

- Temporal logic applies on Kripke structures, i.e. states and transitions
  - each state has properties $\Leftrightarrow$ satisfies atomic formulas
- for Model Checking, the Kripke structure M is readily obtain from a transition system S :
  - forget the label of edges
  - add to each states the properties it verifies
  - add an initial state
- it turns out (Gödel ... again),
  - that this is also a good semantics for intuitionnistic logic,
  - aka natural deduction without excluded-middle !

Example of Kripke Structure



LTL

- LTL is a logic said *linear in time* :
  - **A** very restricted, **E** forbidden
  - $A\varphi$ alowed only if φ contains no A or E **and** A is at the top of the formula
  - "*all the paths verify* φ"
- Examples :
  - $AFGp$ is a LTL formula
  - EFp is not a LTL formula
- Abstract syntax :

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid X\varphi \mid F\varphi \mid G\varphi \mid \varphi U \varphi$$

- Domain of interpretation :
  - paths σ
  - assuming each state s contains a set $I(s)$ of predicates (assumed to be true at s)
  - notations :
    - $\sigma(0)$ is the state at position 0 of σ.
    - σ^k is the truncation of σ starting from position k
- Satisfiability :
  - $\sigma \models p$ iff $p \in I(\sigma(0))$
  - $\sigma \models \neg\varphi$ iff $\sigma \not\models \varphi$
  - $\sigma \models \varphi_1 \vee \varphi_2$ iff $\sigma \models \varphi_1$ or $\sigma \models \varphi_2$
  - $\sigma \models \varphi_1 \wedge \varphi_2$ iff $\sigma \models \varphi_1$ and $\sigma \models \varphi_2$

For the temporal connectives :

- $\sigma \models X\varphi$ iff $\sigma^1 \models \varphi$
- $\sigma \models F\varphi$ iff for some $k \geq 0$, $\sigma^k \models \varphi$
- $\sigma \models G\varphi$ iff for any $k \geq 0, \sigma^k \models \varphi$
- $\sigma \models \varphi_1 U \varphi_2$ iff for some $k \geq 0$ $\sigma^k \models \varphi_2$ and for any $0 \leq i < k$, $\sigma^i \models \varphi_1$
- this is a strong until, since φ_2 must be satisfied in the future.

- duality **F** and **G** : $F\varphi \equiv \neg G \neg \varphi$ and $G\varphi \equiv \neg F \neg \varphi$
- F definable with U : $F\varphi \equiv \texttt{True}\ U\ \varphi$
- commutativity **X**-$\neg$: $\neg X\varphi \equiv X\neg\varphi$
- distributivity : $G(\varphi_1 \wedge \varphi_2) \equiv G\varphi_1 \wedge G\varphi_2$, $F(\varphi_1 \vee \varphi_2) \equiv F\varphi_1 \vee F\varphi_2$
- idempotency : $GG\varphi \equiv G\varphi$, $FF\varphi \equiv F\varphi$
- and many other identities ...

Where is LTL ? Universal Path Quantifier

- we defined validity for path formulas :

$$\varphi_p ::= p \mid \neg\varphi_p \mid \varphi_p \vee \varphi_p \mid \varphi_p \wedge \varphi_p \mid X\varphi_p \mid F\varphi_p \mid G\varphi_p \mid \varphi_p U \varphi_p$$

- and we have only one type of state formulas : $\varphi_s ::= \mathbf{A}\varphi_p$
- for state formulas, the interpretation domain is a pair (M, s)
  - M is a Kripke structure
  - s is a state (initial state, or not)
- $(M, s) \models A\varphi_p$ iff all paths σ starting at s are such that $\sigma \models \varphi_p$
- and $M \models \varphi_s$ iff $(M, s_0) \models \varphi$

CTL and CTL*

- CTL* is a branching-time logic
- where combination of path quantifiers and temporal connectives is unrestricted
- distinction between state and path formulas
  - but beware, that the formulas are now interleaved, no hiearchy
    - state formulas :

$$\varphi_s ::= p \mid \neg\varphi_s \mid \varphi_s \vee \varphi_s \mid \varphi_s \wedge \varphi_s \mid \mathbf{A}\varphi_p \mid \mathbf{E}\varphi_p$$

    - path formulas :

$$\varphi_p ::= \varphi_s \mid \neg\varphi_p \mid \varphi_p \vee \varphi_p \mid \varphi_p \wedge \varphi_p \mid X\varphi_p \mid F\varphi_p \mid G\varphi_p \mid \varphi_p U \varphi_p$$

State Formulas, relation $\models_s$

- $(M, s) \models_s p$ iff $p \in I(s)$
- $(M, s) \models_s A\varphi_p$ iff all paths σ starting at s verify $(M, \sigma) \models_p \varphi$
- $(M, s) \models_s E\varphi_p$ iff for at least one path σ starting at s, $(M, \sigma) \models_p \varphi$

Path Formulas, relation $\models_p$

- same relation as in LTL, except :
- $(M, \sigma) \models_p \varphi_s$ iff $(M, \sigma(0)) \models_s \varphi_s$

Finally, $M \models \varphi$ iff $(M, s_0) \models_s \varphi$

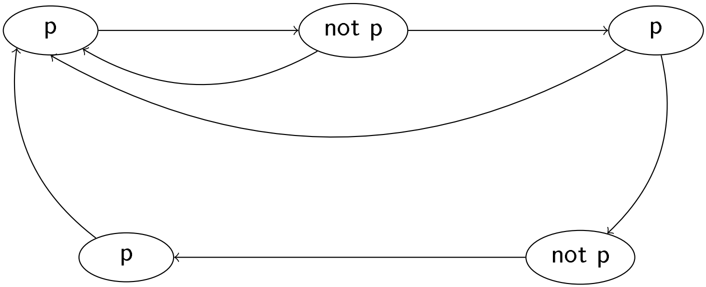
- all the LTL identities
- additional duality **A**-**E** : $\neg E\varphi \equiv A\neg\varphi$ and $\neg A\varphi \equiv E\neg\varphi$
- **Beware !** $A\varphi$ does not necessarily implies $E\varphi$
  - only if there exists a path from the current state

- CTL is a restriction of CTL*
  - can use freely path quantifiers **A**,**E**
  - but temporal connectives **X**,**F**,**G**,**U** must always be preceded by **A** or **E**
- Examples :
  - $EF(AGp)$ is in the CTL language
  - $E(Gp \wedge Xq)$ and $AFGp$ are in CTL* but not in CTL

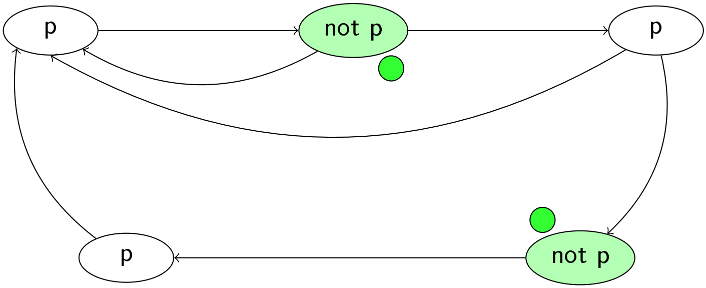
- the semantics of CTL can be given only in termes of states
  - indeed, all the formulas of CTL are state formulas of CTL*
  - ignore paths
  - implies specific and efficient algorithms
- Drawback : CTL does not express fairness
  - adapt the semantics of M
  - Fair CTL, widely used

CTL Model Checking

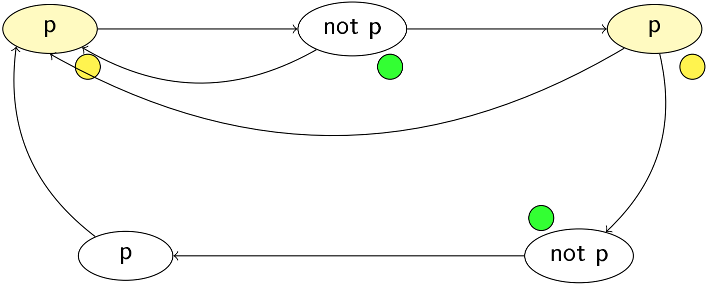
Example : $(\neg EX \neg p) \wedge p$



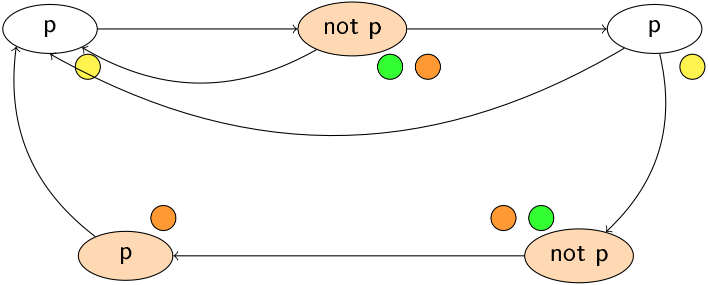
- inductive approach : $\neg p$



- inductive approach : $\neg p$, **EX**$\neg p$

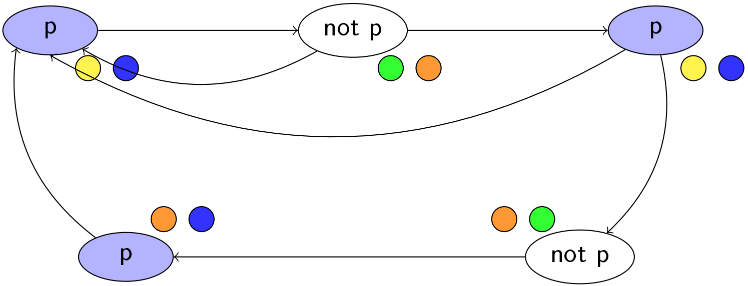


- inductive approach : $\neg p$, $\mathbf{EX}\neg p$, $\neg\mathbf{EX}\neg p$

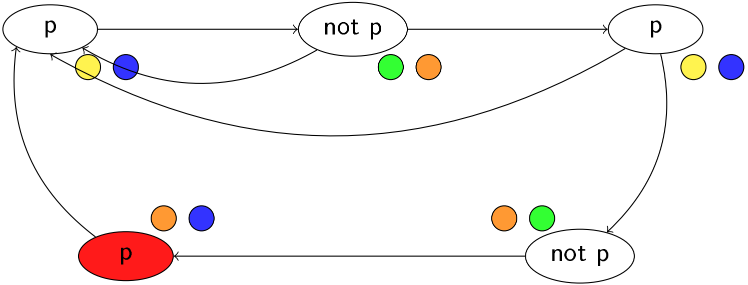


Example : $(\neg EX \neg p) \wedge p$

- inductive approach : $\neg p$, **EX**$\neg p$, $\neg$**EF**$\neg p$, p



- inductive approach : $\neg p$, **EX**$\neg p$, $\neg$**EF**$\neg p$, p and the winner is **blueorange marked**



- Historical importance (Clarke & Emerson, 1981)
- linear in the size of the input (assuming the model has constant-size)
- quite specific to CTL
- reason on states, rather than on paths/execution traces
  - label the states that verify a subformula ψ of φ by ψ itself
  - recursive labelling : with the subformulas first
  - $M \models \varphi$ iff s_0 is labelled with φ

- Basic Operations :
  - q labelled with p if $p \in I(q)$ (given by M)
  - q labelled with $\varphi \wedge \psi$ iff labelled with both φ and ψ
- Case $\mathbf{EX}\varphi$
  - q labelled iff there exists $q \to q'$ s.t. q' labelled with φ
- Case $\mathbf{EF}\varphi$:
  - q labelled iff there exists $q \to \cdots \to q'$ and q' labelled with φ

- let Q_φ the set of states labelled with φ
- we also compute those sets
- Case $\mathbf{E}\varphi\mathbf{U}\psi$
  - $q \in Q_{E\varphi U\psi}$ iff q can reach Q_ψ with a path in Q_φ
- Case $\mathbf{EG}\varphi$:
  - find the nontrivial strongly connected components of Q_φ
  - L is the set of those states
  - $q \in EG\varphi$ iff q can reach L with a path in Q_φ

Complexity of Model Checking in CTL

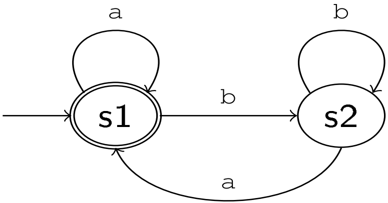
- linear in the size of the formula
- $\mathcal{O}(|M|.|\varphi|)$
- but beware that M is assumed to be a constant
- computing M can be exponential (variables, concurrent processes)

LTL Model Checking

Büchi Automata

An extension of usual automata on infinite words

- $B = \langle \Sigma, Q, \rightarrow, q_o, F \rangle$
- an infinite word is accepted iff w visits *infinitely* many times F



- finite automaton : **words ending with a**
- Büchi automaton : **words having an infinite number of a**

- A tool to recognize sets of infinite words
- $B = \langle \Sigma, Q, \rightarrow, q_0, F \rangle$:
  - Σ, the alphabet (finite number of letters)
  - Q, the set of states
  - $\rightarrow \subseteq Q \times \Sigma \times Q$, the transitions
  - q_0 the initial state
  - $F \subseteq Q$ the final/accepting states
- an infinite word w is recognized by B iff its execution :

$$q_0 \stackrel{w_0}{\rightarrow} q_1 \stackrel{w_1}{\rightarrow} \cdots \stackrel{w_n}{\rightarrow} q_{n+1} \stackrel{w_{n+1}}{\rightarrow} \cdots$$

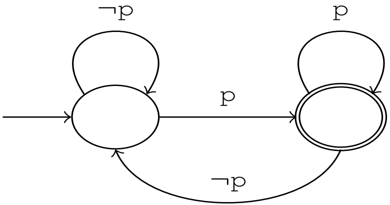
is such that for an infinite set of indices $i_1 < \cdots < i_n < \cdots$, $q_{i_k} \in F$

- define sets of infinite traces
  - $\mathcal{L}(B)$ is the language recognized by B
  - those are called ω-regular languages
- operation on those languages
  - test of emptiness : is a strongly connected component intersecting F accessible from q_0 ?
  - intersection : compute $B_\otimes$ such that $\mathcal{L}(B_\otimes) = \mathcal{L}(B_1) \cap \mathcal{L}(B_2)$
  - complementation very hard to implement $\mathcal{O}(2^{n^2})$ at best, $\mathcal{O}(2^{2^n})$ in practice

- a Kripke structure M is the same as an automaton, it defines a language of infinite traces (paths)
- if we have another automaton B_{bad} defining a language $\mathcal{L}(B_{bad})$, would be easy to check emptiness of $\mathcal{L}(B_M) \cap \mathcal{L}(B_{bad})$
- can we transform a LTL formula φ into a Büchi automaton such that $[\![\neg\varphi]\!] = \mathcal{L}(B_{\neg\varphi})$?
- if this is the case, then :
  1. Transform M into B_M (immediate)
  2. Transform φ_p into $B_{\neg\varphi_p}$
  3. Compute $B_\otimes = B_M \otimes B_{\neg\varphi_p}$
  4. test $\mathcal{L}(B_\otimes) = \emptyset$

Theorem

It is possible to translate any LTL formula into a Büchi automaton defining the same language



On the **AGF**p example

- double-exponential complexity if we use complementation