OUTLINE

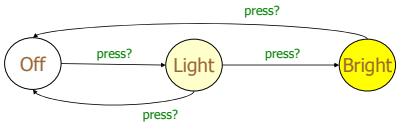
- Introduction
  - Lecture 1: Motivation, examples, problems to solve
- Modeling and Verication of Timed Systems
  - Lecture 2: Timed automata, and timed automata in UPPAAL
  - Lecture 3: Symbolic verification: the core of UPPAAL
  - Lecture 4: Verification Options in UPPAAL
- Towards a Unified Framework
  - Lecture 5: Modeling, verification, real time scheduling, code synthesis

From UPPAAL to TIMES

Modeling Real Time Systems

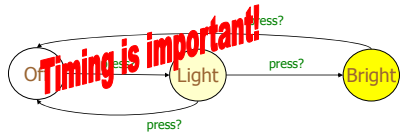
Timed automata: Syntax and Semantics

Intelligent Light Control



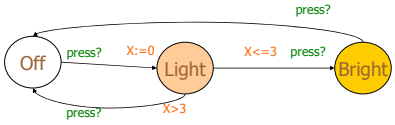
WANT: if press is issued twice quickly
then the light will get brighter; otherwise the light is turned off.

Intelligent Light Control



WANT: if press is issued twice quickly
then the light will get brighter; otherwise the light is turned off.

Intelligent Light Control (with timer)

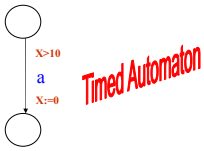


Solution: Add real-valued clock x

Alur and Dill 1990

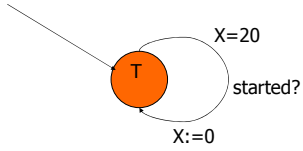
Timed Automata = Finite Automata + Clocks

Timed Automata: Syntax

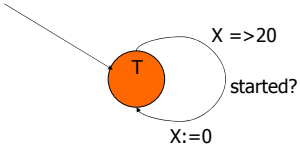


- Guard
  - Timing constraints e.g. X>10
- Action
  - Synchronization e.g. a
- Clock reset
  - Reset clock to 0 e.g. X:=0

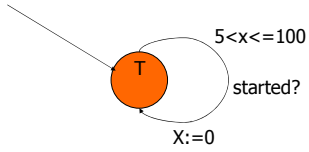
Timed Automata: Example
(periodic task)



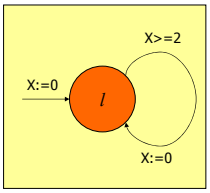
Timed Automata: Example
(sporadic task)



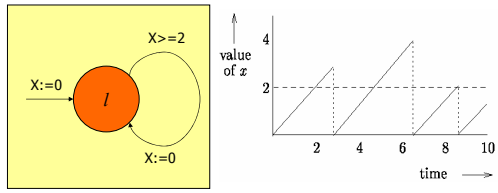
Timed Automata: Example
(aperiodic task)



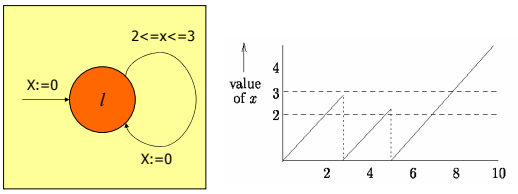
Timed Automata: Example



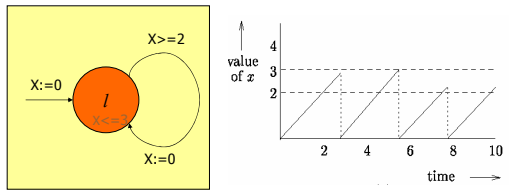
Timed Automata: Example



Timed Automata: Example



Timed Automata: Example



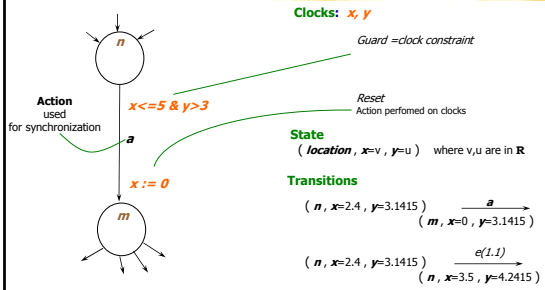
Clock Constraints

For set C of clocks with $x, y \in C$, the set of *clock constraints* over C, $\Psi(C)$, is defined by

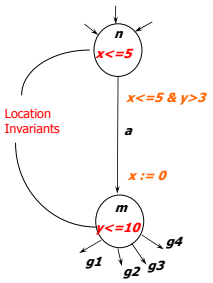
$$\alpha ::= x \prec c \mid x - y \prec c \mid \neg\alpha \mid (\alpha \wedge \alpha)$$

where $c \in \mathbb{N}$ and $\prec \in \{<, \leqslant\}$.

Timed Automata: Semantics



Timed Automata with *Invariants*



Clocks: *x, y*

Transitions

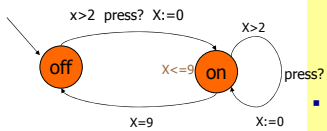
(*n* , *x*=2.4 , *y*=3.1415) ~~e(2.2)~~

(*n* , *x*=2.4 , *y*=3.1415) e(1.1)

(*n* , *x*=3.5 , *y*=4.2415)

Invariants insure progress!!

Timed Automata: Light Switch

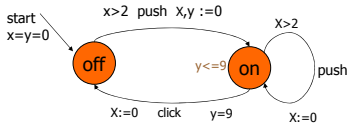


- Switch may be turned on whenever at least 2 time units has elapsed since last "turn off"
- Light automatically switches off after 9 time units if it is not pressed.

Semantics (definition)

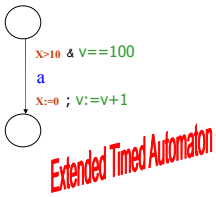
- *clock valuations*: $V(C) \quad v : C \to R_{\geq 0}$
- *state*: $(l, v) \quad where \quad l \in L \ and \ v \in V(C)$
- *action transition* $(l,v) \xrightarrow{a} (l',v') \ iff \ l \xrightarrow{g\ a\ r} l'$
  $g(v) \ and \ v' = v[r] \ and \ Inv(l')(v')$
- *delay Transition* $(l,v) \xrightarrow{d} (l, v+d) \ iff$
  $Inv(l)(v+d') \ whenever \ d' \leq d \in R_{\geq 0}$

Timed Automata: Example



$(off, x = y = 0) \xrightarrow{3.5} (off, x = y = 3.5) \xrightarrow{push}$
$(on, x = y = 0) \xrightarrow{\pi} (on, x = y = \pi) \xrightarrow{push}$
$(on, x = 0, y = \pi) \xrightarrow{3} (on, x = 3, y = \pi + 3) \xrightarrow{9-(\pi+3)}$
$(on, x = 9 - (\pi + 3), y = 9) \xrightarrow{click} (off, x = 0, y = 9) \ldots$

Timed Automata in UPPAAL



- Events
  - synchronization
  - interrupts
- Timing constraints
  - specifying event arrivals
  - e.g. Periodic and sporadic
- Data variables
  - Guards (C code)
  - Assignments (C code)

Extended Timed Automata in UPPAAL

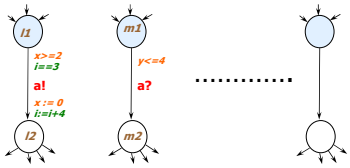
Timed automata
+ data types (integers, arrays etc)
+ concurrency (synchronous communication from CCS, urgent actions, committed locations)

Extended Timed Automata in UPPAAL

Timed automata
+ data types (integers, arrays etc)
+ concurrency (synchronous communication from CCS, urgent actions, committed locations)

i.e. **Networks of Timed Automata**

Networks of Timed Automata + Integer Variables +....



Two-way synchronization on *complementary* actions.

Closed Systems!

Example transitions

(*l1, m1,*....., x=2, y=3.5, i=3,.....) $\xrightarrow{tau}$ (*l2,m2,*.....,x=0, y=3.5, i=7,.....)