Verification

Lecture 20

Bernd Finkbeiner

UNIVERSITÄT
DES
SAARLANDES

Plan for today

- Timed automata
- UPPAAL

(more on simulation tomorrow.)

Time-critical systems

- Timing issues are of crucial importance for many systems, e.g.,
  - landing gear controller of an airplane, railway crossing, robot controllers
  - steel production controllers, communication protocols . . . . . .
- In time-critical systems correctness depends on:
  - not only on the logical result of the computation, but
  - also on the time at which the results are produced
- How to model timing issues:
  - discrete-time or continuous-time?

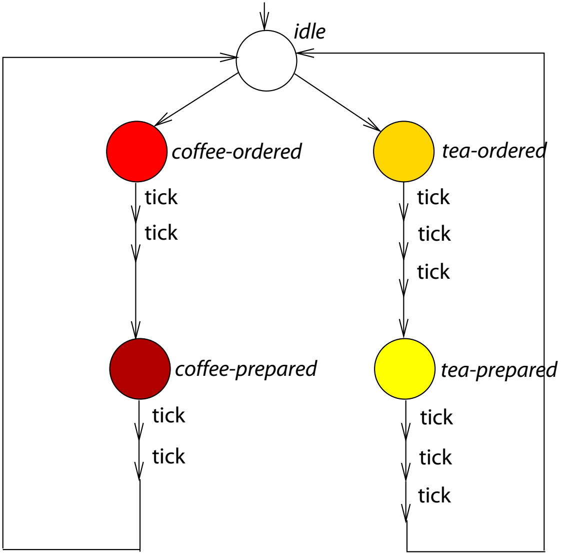
A discrete time domain

- Time has a discrete nature, i.e., time is advanced by discrete steps
  - time is modelled by naturals; actions can only happen at natural time values
  - a specific tick action is used to model the advance of one time unit
  - $\Rightarrow$ delay between any two events is always a multiple of the minimal delay of one time unit
- Properties can be expressed in traditional temporal logic
  - the next-operator "measures" time
  - two time units after being red, the light is green: $\mathrm{G}\,(\mathit{red} \Rightarrow \mathrm{X}\,\mathrm{X}\,\mathit{green})$
  - within two time units after red, the light is green:

$$\mathrm{G}\,(\mathit{red} \Rightarrow (\mathit{green} \vee \mathrm{X}\,\mathit{green} \vee \mathrm{X}\,\mathrm{X}\,\mathit{green}))$$

- Main application area: synchronous systems, e.g., hardware

A discrete-time coffee machine

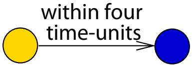


A discrete time domain

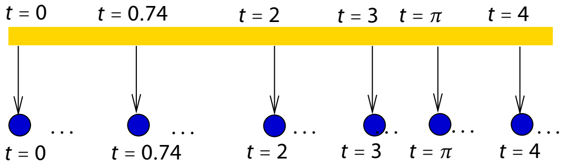
- Main advantage: conceptual simplicity
  - state graphs systems equipped with a "tick" transition suffice
  - standard temporal logics can be used
  - $\Rightarrow$ traditional model-checking algorithms suffice
- Main limitations:
  - (minimal) delay between any pair of actions is a multiple of an a priori fixed minimal delay
  - $\Rightarrow$ difficult (or impossible) to determine this in practice
  - $\Rightarrow$ limits modeling accuracy
  - $\Rightarrow$ inadequate for asynchronous systems. e.g., distributed systems

A continuous time-domain

If time is continuous, state changes can happen at any point in time:



is modeled by



but: infinitely many states and infinite branching

How to check a property like:

once in a yellow state, eventually the system is in a blue state within π time-units?

Approach

- Restrict expressivity of the property language
  - e.g., only allow reference to natural time units

$\Longrightarrow$ Timed CTL

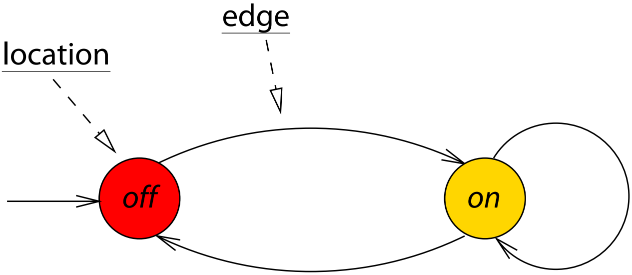
- Model timed systems symbolically rather than explicitly

$\Longrightarrow$ Timed Automata

- Consider a finite quotient of the infinite state space on-demand
  - i.e., using an equivalence that depends on the property and the timed automaton

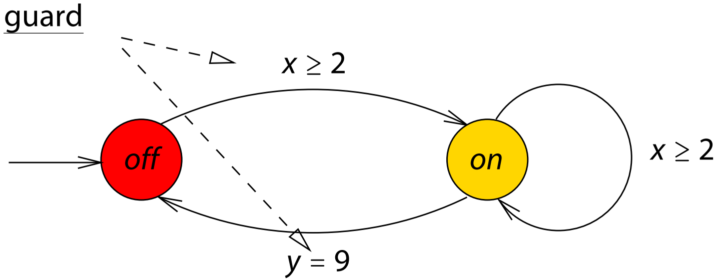
$\Longrightarrow$ Region Automata

What is a timed automaton?



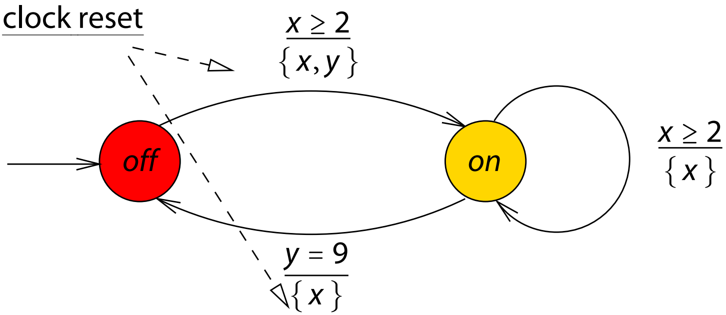
- a program graph with locations and edges
- a location is labeled with the valid atomic propositions
- taking an edge is instantaneous, i.e, consumes no time

What is a timed automaton?



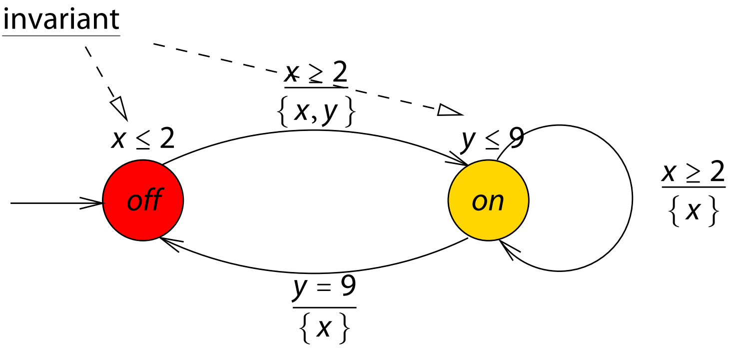
- equipped with real-valued clocks $x, y, z, \ldots$
- clocks advance implicitly, all at the same speed
- logical constraints on clocks can be used as guards of actions

What is a timed automaton?



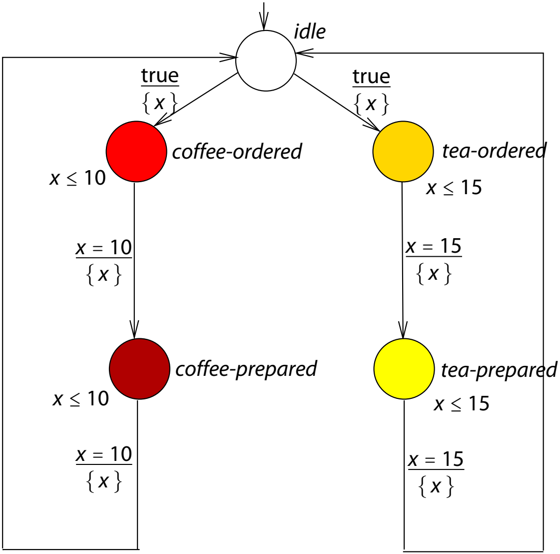
- clocks can be reset when taking an edge
- assumption:
  all clocks are zero when entering the initial location initially

What is a timed automaton?



- guards indicate when an edge may be taken
- a location invariant specifies the amount of time that may be spent in a location
  - before a location invariant becomes invalid, an edge must be taken

A real-time coffee machine



Clock constraints

- Clock constraints over set C of clocks are defined by:

$$g ::= \text{true} \mid x < c \mid x - y < c \mid x \leq c \mid x - y \leq c \mid \neg g \mid g \wedge g$$

  - where $c \in \mathbb{N}$ and clocks $x, y \in C$
  - rational constants would do; neither reals nor addition of clocks!
  - let $CC(C)$ denote the set of clock constraints over C
  - shorthands: $x \geq c$ denotes $\neg(x < c)$ and $x \in [c_1, c_2)$ or $c_1 \leq x < c_2$ denotes $\neg(x < c_1)$ & $(x < c_2)$
- Atomic clock constraints do not contain true, $\neg$ and $\wedge$
  - let $ACC(C)$ denote the set of atomic clock constraints over C
- Simplification: In the following, we assume constraints are diagonal-free, i.e., do neither contain $x - y \leq c$ nor $x - y < c$.

Timed automaton

A timed automaton is a tuple

$$TA = \left(Loc, Act, C, \rightsquigarrow, Loc_0, inv, AP, L\right) \quad \text{where:}$$

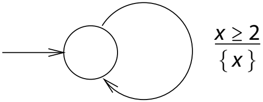
- Loc is a finite set of locations.
- $Loc_0 \subseteq Loc$ is a set of initial locations
- C is a finite set of clocks
- $L : Loc \to 2^{AP}$ is a labeling function for the locations
- $\rightsquigarrow \subseteq Loc \times CC(C) \times Act \times 2^C \times Loc$ is a transition relation, and
- $inv : Loc \to CC(C)$ is an invariant-assignment function

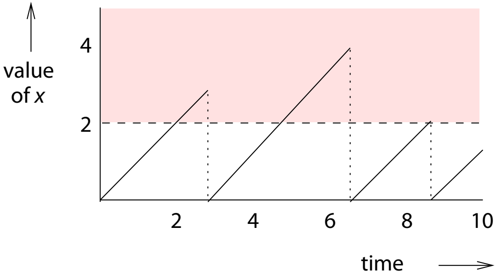
Intuitive interpretation

- Edge $\ell \xrightarrow{g:\alpha,C'} \ell'$ means:
  - action α is enabled once guard g holds
  - when moving from location ℓ to ℓ', any clock in C' will be reset to zero
- $inv(\ell)$ constrains the amount of time that may be spent in location ℓ
  - the location ℓ must be left before the invariant $inv(\ell)$ becomes invalid

Guards versus location invariants

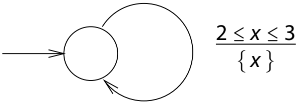
The effect of a lowerbound guard:

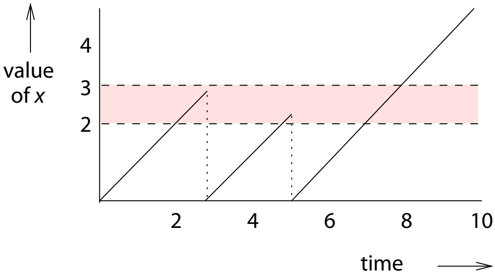




Guards versus location invariants

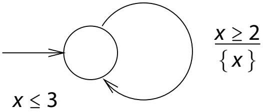
The effect of a lowerbound and upperbound guard:

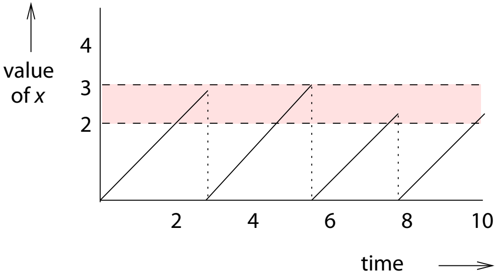




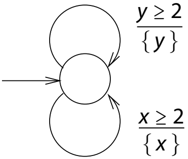
Guards versus location invariants

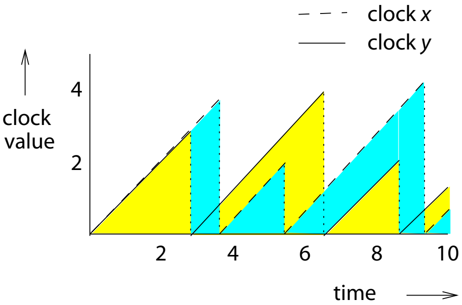
The effect of a guard and an invariant:





Arbitrary clock differences





Composing timed automata

Let $TA_i \;=\; \big(Loc_i, Act_i, C_i, \rightsquigarrow_i, Loc_{0,i}, inv_i, AP, L_i\big)$ and H an action-set

$$TA_1 \parallel_H TA_2 = \big(Loc, Act_1 \cup Act_2, C, \rightsquigarrow, Loc_0, inv, AP, L\big) \quad \text{where:}$$

- $Loc = Loc_1 \times Loc_2$ and $Loc_0 = Loc_{0,1} \times Loc_{0,2}$ and $C = C_1 \cup C_2$
- $inv(\langle \ell_1, \ell_2 \rangle) = inv_1(\ell_1) \;\wedge\; inv_2(\ell_2)$ and $L(\langle \ell_1, \ell_2 \rangle) = L_1(\ell_1) \cup L_2(\ell_2)$
- $\rightsquigarrow$ is defined by the inference rules:

for $\alpha \in H$ $$\frac{\ell_1 \overset{g_1:\alpha,D_1}{\rightsquigarrow_1} \ell_1' \;\wedge\; \ell_2 \overset{g_2:\alpha,D_2}{\rightsquigarrow_2} \ell_2'}{\langle \ell_1, \ell_2 \rangle \overset{g_1 \wedge g_2:\alpha,D_1 \cup D_2}{\rightsquigarrow} \langle \ell_1', \ell_2' \rangle}$$

for $\alpha \notin H$: $$\frac{\ell_1 \overset{g:\alpha,D}{\rightsquigarrow_1} \ell_1'}{\langle \ell_1, \ell_2 \rangle \overset{g:\alpha,D}{\rightsquigarrow} \langle \ell_1', \ell_2 \rangle} \quad \text{and} \quad \frac{\ell_2 \overset{g:\alpha,D}{\rightsquigarrow_2} \ell_2'}{\langle \ell_1, \ell_2 \rangle \overset{g:\alpha,D}{\rightsquigarrow} \langle \ell_1, \ell_2' \rangle}$$

Clock valuations

- A clock valuation v for set C of clocks is a function $v : C \longrightarrow \mathbb{R}_{\geq 0}$
  - assigning to each clock $x \in C$ its current value $v(x)$
- Clock valuation $v+d$ for $d \in \mathbb{R}_{\geq 0}$ is defined by:
  - $(v+d)(x) = v(x) + d$ for all clocks $x \in C$
- Clock valuation reset x in v for clock x is defined by:

$$(\text{reset } x \text{ in } v)(y) = \begin{cases} v(y) & \text{if } y \neq x \\ 0 & \text{if } y = x. \end{cases}$$

  - reset x in (reset y in v) is abbreviated by reset x, y in v

Timed automaton semantics

For timed automaton $TA = \big(Loc, Act, C, \rightsquigarrow, Loc_0, inv, AP, L\big)$:
Transition system $TS(TA) = (S, Act', \rightarrow, I, AP', L')$ where:

- $S = Loc \times val(C)$, state $s = \langle \ell, v \rangle$ for location ℓ and clock valuation v
- $Act' = Act \cup \mathbb{R}_{\geq 0}$, (discrete) actions and time passage actions
- $I = \{ \langle \ell_0, v_0 \rangle \mid \ell_0 \in Loc_0 \ \wedge \ v_0(x) = 0$ for all $x \in C \}$
- $AP' = AP \cup ACC(C)$
- $L'(\langle \ell, v \rangle) = L(\ell) \cup \{ g \in ACC(C) \mid v \models g \}$
- $\rightarrow$ is the transition relation defined on the next slide

Timed automaton semantics

The transition relation $\rightarrow$ is defined by the following two rules:

- Discrete transition: $\langle \ell, v \rangle \xrightarrow{d} \langle \ell', v' \rangle$ if all following conditions hold:
  - there is an edge labeled $(g : \alpha, D)$ from location ℓ to ℓ' such that:
  - g is satisfied by v, i.e., $v \models g$
  - $v' = v$ with all clocks in D reset to 0, i.e., $v' =$ reset D in v
  - v' fulfills the invariant of location ℓ', i.e., $v' \models inv(\ell')$
- Delay transition: $\langle \ell, v \rangle \xrightarrow{\alpha} \langle \ell, v{+}d \rangle$ for positive real d
  - if for any $0 \leq d' \leq d$ the invariant of ℓ holds for $v{+}d'$, i.e. $v{+}d' \models inv(\ell)$

Time divergence

- Let for any $t < d$, for fixed $d \in \mathbb{R}_{>0}$, clock valuation $\eta + t \models inv(\ell)$
- A possible execution fragment starting from the location ℓ is:

$$\langle \ell, \eta \rangle \xrightarrow{d_1} \langle \ell, \eta + d_1 \rangle \xrightarrow{d_2} \langle \ell, \eta + d_1 + d_2 \rangle \xrightarrow{d_3} \langle \ell, \eta + d_1 + d_2 + d_3 \rangle \xrightarrow{d_4} \ldots$$

  - where $d_i > 0$ and the infinite sequence $d_1 + d_2 + \ldots$ converges towards d
  - such path fragments are called time-convergent

  $\Rightarrow$ time advances only up to a certain value
- Time-convergent execution fragments are unrealistic and ignored
  - much like unfair paths (as we will see later on)

Time divergence

- Infinite path fragment π is time-divergent if $ExecTime(\pi) = \infty$
- The function $ExecTime : Act \cup \mathbb{R}_{>0} \rightarrow \mathbb{R}_{\geq 0}$ is defined as:

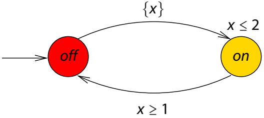
$$ExecTime(\tau) = \begin{cases} 0 & \text{if } \tau \in Act \\ d & \text{if } \tau = d \in \mathbb{R}_{>0} \end{cases}$$

- For infinite execution fragment $\rho = s_0 \xrightarrow{\tau_1} s_1 \xrightarrow{\tau_2} s_2 \ldots$ in $TS(TA)$ let:

$$ExecTime(\rho) = \sum_{i=0}^{\infty} ExecTime(\tau_i)$$

  - for path fragment π in $TS(TA)$ induced by ρ: $ExecTime(\pi) = ExecTime(\rho)$
- For state s in $TS(TA)$: $Paths_{div}(s) = \{ \pi \in Paths(s) \mid \pi \text{ is time-divergent} \}$

Example: light switch



The path π in $TS(Switch)$ in which on- and of-periods of one minute alternate:

$$\pi = \langle off, 0\rangle\, \langle off, 1\rangle\, \langle on, 0\rangle\, \langle on, 1\rangle\, \langle off, 1\rangle\, \langle off, 2\rangle\, \langle on, 0\rangle\, \langle on, 1\rangle\, \langle off, 1\rangle \ldots$$

is time-divergent as $ExecTime(\pi) = 1 + 1 + 1 + \ldots = \infty$.
The path:

$$\pi' = \langle off, 0\rangle\, \langle off, 1/2\rangle\, \langle off, 3/4\rangle\, \langle off, 7/8\rangle\, \langle off, 15/16\rangle \ldots$$

is time-convergent, since $ExecTime(\pi') = \sum_{i \geq 1} \left(\frac{1}{2}\right)^i = 1 < \infty$

Timelock

- State $s \in TS(TA)$ contains a timelock if $Paths_{div}(s) = \varnothing$
  - there is no behavior in s where time can progress ad infinitum
  - clearly: any terminal state contains a timelock (but also non-terminal states may contain a timelock)
  - terminal location does not necessarily yield a state with timelock (e.g., inv = true)
- TA is timelock-free if no state in $Reach(TS(TA))$ contains a timelock
- Timelocks are considered as modeling flaws that should be avoided

Zenoness

- A TA that performs infinitely many actions in finite time is Zeno
- Path π in $TS(TA)$ is Zeno if:
  - it is time-convergent, and
  - infinitely many actions $\alpha \in Act$ are executed along π
- TA is non-Zeno if there does not exist an initial Zeno path in $TS(TA)$
  - any π in $TS(TA)$ is time-divergent or
  - is time-convergent with nearly all (i.e., all except for finitely many) transitions being delay transitions
- Zeno paths are considered as modeling flaws that should be avoided

A sufficient criterion for Non-Zenoness

Let *TA* with set C of clocks such that for every control cycle:

$$\ell_0 \overset{g_1:\alpha_1,C_1}{\rightsquigarrow} \ell_1 \overset{g_2:\alpha_2,C_2}{\rightsquigarrow} \ldots \overset{g_n:\alpha_n,C_n}{\rightsquigarrow} \ell_n$$

there exists a clock $x \in C$ such that:

1. $x \in C_i$ for some $0 < i \leq n$, and
2. there exists a constant $c \in \mathbb{N}_{>0}$ such that for all clock evaluations η:

$$\eta(x) < c \text{ implies } (\eta \not\models g_j \text{ or } \eta \not\models inv(\ell_j)), \text{ for some } 0 < j \leq n$$

Then: *TA* is non-Zeno

Timelock, time-divergence and Zenoness

- A timed automaton is only considered an adequate model of a time-critical system if it is:

  non-Zeno and timelock-free

- Time-convergent paths will be explicitly excluded from the analysis.

Timed CTL

Syntax of TCTL state-formulas over *AP* and set *C*:

$$\Phi ::= \text{true} \;\Big|\; a \;\Big|\; g \;\Big|\; \Phi \wedge \Phi \;\Big|\; \neg\Phi \;\Big|\; \mathsf{E}\,\varphi \;\Big|\; \mathsf{A}\,\varphi$$

where $a \in AP$, $g \in ACC(C)$ and φ is a path-formula defined by:

$$\varphi ::= \Phi\, \mathsf{U}^{J}\, \Phi$$

where $J \subseteq \mathbb{R}_{\geq 0}$ is an interval whose bounds are naturals
Forms of J: $[n,m]$, $(n,m]$, $[n,m)$ or (n,m) for $n,m \in \mathbb{N}$ and $n \leq m$

for right-open intervals, $m = \infty$ is also allowed

Some abbreviations

- $F^J \Phi = \text{true}\, U^J\, \Phi$
- $EG^J\, \Phi = \neg A F^J \neg \Phi$ and $AG^J\, \Phi = \neg E F^J \neg \Phi$
- $F\ \Phi = F^{[0,\infty)}\, \Phi$ and $G\ \Phi = G^{[0,\infty)}\, \Phi$

Semantics of TCTL

For state $s = \langle \ell, \eta \rangle$ in $TS(TA)$ the satisfaction relation $\models$ is defined by:

$s \models \text{true}$		
$s \models a$	iff	$a \in L(\ell)$
$s \models g$	iff	$\eta \models g$
$s \models \neg\Phi$	iff	not $s \models \Phi$
$s \models \Phi \wedge \Psi$	iff	$(s \models \Phi)$ and $(s \models \Psi)$
$s \models \mathrm{E}\,\varphi$	iff	$\pi \models \varphi$ for some $\pi \in Paths_{div}(s)$
$s \models \mathrm{A}\,\varphi$	iff	$\pi \models \varphi$ for all $\pi \in Paths_{div}(s)$

path quantification over time-divergent paths only

The $\Longrightarrow$ relation

- For infinite path fragments in $TS(TA)$ performing ∞ many actions let:

$$s_0 \xRightarrow{d_0} s_1 \xRightarrow{d_1} s_2 \xRightarrow{d_2} \ldots \qquad \text{with } d_0, d_1, d_2 \ldots \geq 0$$

denote the equivalence class containing all infinite path fragments induced by execution fragments of the form:

$$s_0 \underbrace{\xrightarrow{d_0^1} \ldots \xrightarrow{d_0^{k_0}}}_{\substack{\text{time passage of} \\ d_0 \text{ time-units}}} s_0{+}d_0 \xrightarrow{\alpha_1} s_1 \underbrace{\xrightarrow{d_1^1} \ldots \xrightarrow{d_1^{k_1}}}_{\substack{\text{time passage of} \\ d_1 \text{ time-units}}} s_1{+}d_1 \xrightarrow{\alpha_2} s_2 \underbrace{\xrightarrow{d_2^1} \ldots \xrightarrow{d_2^{k_2}}}_{\substack{\text{time passage of} \\ d_2 \text{ time-units}}} s_2{+}d_2 \xrightarrow{\alpha_3} \ldots$$

where $k_i \in \mathbb{N}$, $d_i \in \mathbb{R}_{\geq 0}$ and $\alpha_i \in Act$ such that $\sum_{j=1}^{k_i} d_i^j = d_i$.
Notation: $s_i{+}d = \langle \ell_i, \eta_i{+}d \rangle$ where $s_i = \langle \ell_i, \eta_i \rangle$.

- For infinite path fragments in $TS(TA)$ performing finitely many actions:

$$s_0 \xRightarrow{d_0} s_1 \xRightarrow{d_1} s_2 \xRightarrow{d_2} \ldots \xRightarrow{d_{n-1}} s_n \xRightarrow{1} s_{n+1} \xRightarrow{1} s_{n+2} \xRightarrow{1} \ldots$$

Semantics of TCTL

For time-divergent path $\pi \in s_0 \xRightarrow{d_0} s_1 \xRightarrow{d_1} \ldots$:

$\pi \models \Phi\, \mathsf{U}^{J}\, \Psi$

iff

$\exists\, i \geq 0.\ s_i{+}d \models \Psi$ for some $d \in [0, d_i]$ with $\sum_{k=0}^{i-1} d_k + d \in J$
and
$\forall j \leq i.\ s_j{+}d' \models \Phi \vee \Psi$ for every $d' \in [0, d_j]$ with $\sum_{k=0}^{j-1} d_k + d' \leq \sum_{k=0}^{i-1} d_k + d$