On Zone-Based Reachability Computation for Duration-Probabilistic Automata

How the Timed Automaton Lost its Tail

Oded Maler

CNRS - VERIMAG
Grenoble, France

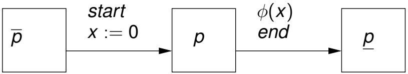
2010

Joint work with Kim Larsen (Aalborg) and Bruce Krogh (CMU)

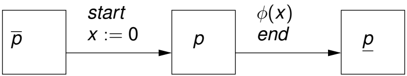
Summary

- Processes that take time
- Worst-case versus average case reasoning about time
- Duration probabilistic automata
- Forward reachability and density transformers
- Concluding remarks

Processes that Take Time

- We are interested in processes that take some time to conclude after having started
- Can model almost anything:
  - Transmission delays in a network
  - Propagation delays in digital gates
  - Execution time of programs
  - Duration of a production step in a factory
  - Time to produce proteins in a cell
  - Cooking recipes
  - Project planning
- Mathematically they are simple timed automata:
- 

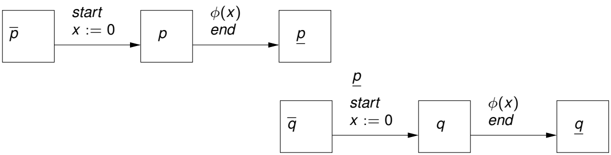
Processes that Take Time

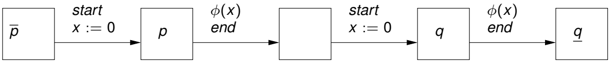
- 

- A waiting state $\overline{p}$;
- A **start** transition which resets a clock x to measure time elapsed in active state p
- An **end** transition guarded by a temporal condition $\phi(x)$
- Condition ϕ can be
  - **true** (no constraint)
  - $x = d$ (deterministic)
  - $x \in [a, b]$ (non-deterministic)
  - Probabilistically distributed

Composition

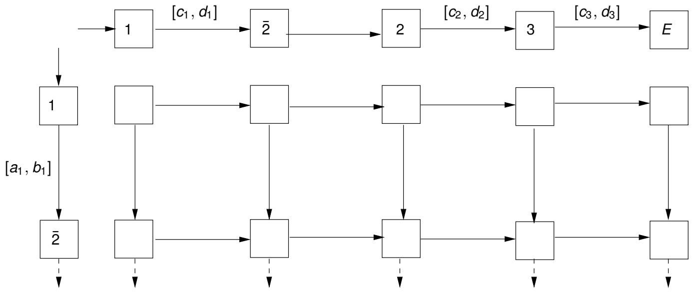
- Such processes can be combined:
- Sequentially, to represent precedence relations between tasks, for example p precedes q:





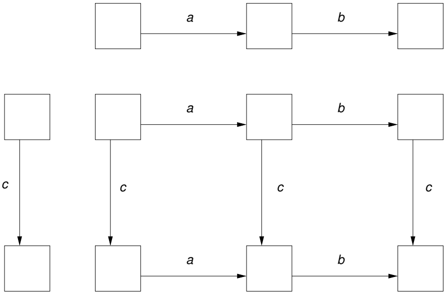
Composition

- Such processes can be combined:
- In parallel, to express partially-independent processes, sometimes competing with each other



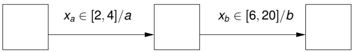
Levels of Abstraction: Untimed

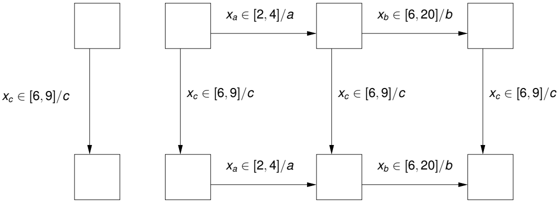
- Consider two parallel processes, one doing $a \cdot b$ and the other doing c
- Untimed (asynchronous) modeling assumes nothing concerning duration
- Each process may take between zero and infinity time
- Consequently any interleaving in $(a \cdot b)||c$ is possible



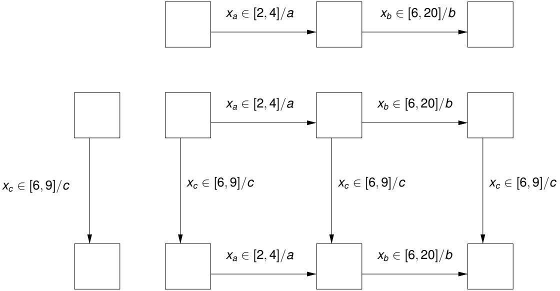
Levels of Abstraction: Timed

- Timed automata and similar formalisms add more detail
- Assume a (positive) lower bound and (finite) upper bound for the duration of each processing step



- 

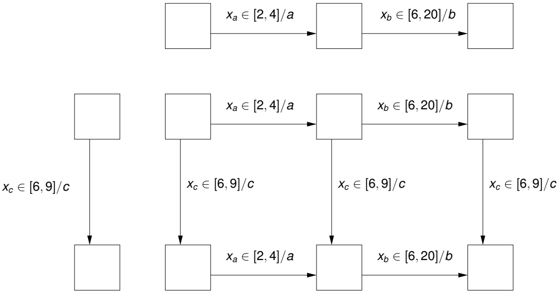
Levels of Abstraction: Timed



- The arithmetics of time eliminates some paths:
- Since $4 < 6$, a must precede c and the set of possible paths is reduced to $a \cdot (b||c) = abc + acb$
- But how likely is abc to occur?

Possible but Unlikely

- How likely is *abc* to occur?



- Each run corresponds to a point in the duration space

$$(y_a, y_b, y_c) \in Y = [2,4] \times [6,20] \times [6,9]$$

- Event *b* precedes *c* only when $y_a + y_b < y_c$
- Since $y_a + y_b$ ranges in $[8,24]$ and $y_c \in [6,9]$, this is less likely than *c* preceding *b*

Levels of Abstraction: Probabilistic Timed

- Interpreting temporal guards probabilistically
- This gives precise **quantitative** meaning to this intuition
- It allows us to:
  - Compute probabilities of different paths (equivalence classes of qualitative behaviors)
  - Compute and compare the **expected performance** of **schedulers**, for example for job-shop problems with probabilistic step durations
  - **Discard low-probability paths** in verification and maybe reduce some of the state and clock explosion

Levels of Abstraction: Probabilistic Timed

- Of course, **continuous-time stochastic processes** are **not** our invention
- But, surprisingly(?), computing these probabilities for such composed processes has rarely been attempted
- Some work in the probabilistic verification community deals with the **very special case** of exponential (memoryless) distribution
- With this distribution, the time that has already elapsed since **start** does not influence the probability over the remaining time to termination
- Notable exceptions:
  - Alur and Bernadsky (GSMP)
  - Vicario et al (stochastic PN)

Probabilistic Interpretation of Timing Uncertainty

- We interpret a duration interval $[a, b]$ as a uniform distribution: all values in the interval are equally likely
- This is expressed via density function

$$\phi(y) = \begin{cases} 1/(b-a) & \text{if } a \leq y \leq b \\ 0 & \text{otherwise} \end{cases}$$

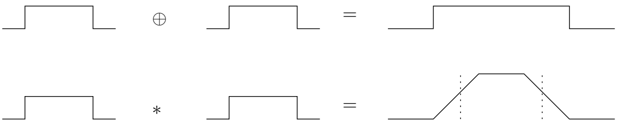
- Interval $[a, b]$ is the **support** of ϕ
- The probability that the actual duration is in some interval $[c, d]$ is

$$P([c, d]) = \int_c^d \phi(\tau) d\tau$$

Minkowski Sum vs. Convolution

- Consider two processes with durations in $[a, b]$ and $[a', b']$ that execute sequentially
- Their total duration is inside the Minkowski sum $[a, b] \oplus [a', b'] = [a + a', b + b']$
- This is what timed automata will compute for you
- With the intervals interpreted as uniform distributions ϕ, ϕ' the total duration is distributed as their convolution

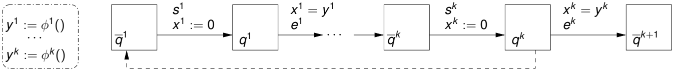
$$\phi * \phi'(y) = \int \phi(y - \tau)\phi'(\tau)d\tau$$



Duration Probabilistic Automata

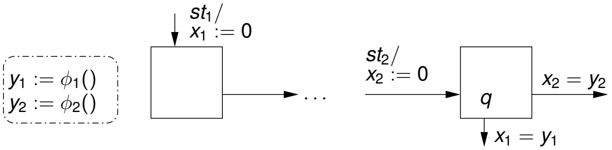
- Duration probabilistic automata (DPA) consist of a composition of simple DPA (SDPA) and a scheduler

$$\mathcal{A} = \mathcal{A}^1 \circ \mathcal{A}^2 \circ \cdots \circ \mathcal{A}^n \circ S$$

- SDPA: (acyclic) alternations of waiting and active states
- 

- The y variables are “static” random variable drawn uniformly from the duration space
- The x variables are clocks reset to zero upon **start** transitions and compared to y upon **end** transitions
- The scheduler issues the **start** transitions

Clocks in Timed Automata and DPA

- A global state of a DPA is a tuple consisting of local states, some active and some inactive (waiting)
- For each active component, its corresponding clock measures the time since its **start** transition
- The clock values determine which **end** transition can be taken from this state (and in which probability)
- In other words, which active processes can **“win the race”**
- 

Zones, Symbolic States and Forward Reachability

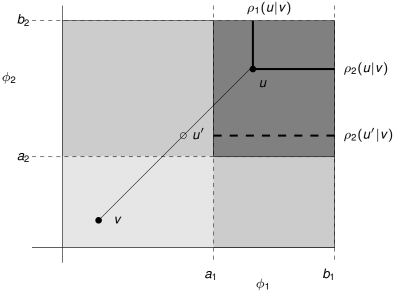
- In timed automata the possible paths are computed using **forward reachability** over symbolic states
- A symbolic state is (q, Z) where q is a global state and Z is a set of clock valuations with which it is possible to reach q
- The reachability tree/graph is constructed iteratively from the initial state using a **successor operator**
- For every transition δ from q to q' the successor operator $Post_\delta$ is defined as:
- $(q', Z') = Post_\delta(q, Z')$ if Z' is the set of clock valuations possible at q' given that Z is the set of possible clock valuations at q

Forward Reachability for DPA

- We adapt this idea to DPA using extended symbolic states of the form (q, Z, ψ) where ψ is a (partial) density over the clock values
- Symbolic state (q', Z', ψ') is a successor of (q, Z, ψ) if:
- Given density ψ on the clock values upon entering q, the density upon taking the transition to q' is ψ'
- The successor operator is a **density transformer** which for **start** transitions is rather straightforward
- The crucial point is how to compute it in a state admitting several competing **end** transitions

Intuition

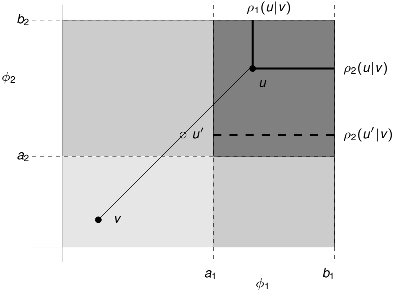
- Consider state q with two competing processes with durations distributed uniformly $\phi_1 = [a_1, b_1]$, $\phi_2 = [a_2, b_2]$
- What is the probability $\rho_i(u|v)$ that transition i wins at some clock valuation $u = (u_1, u_2)$, i.e., $u_i = y_i$, **given** the state was entered at some v?
- First, this probability is non-zero only if v is a **time-predecessor** of u, $v \in \pi(u)$



-

Intuition

- For transition 1 to win, process 1 should choose duration u_1 while process 2 chooses some $y_2 > u_2$
- Thus $\rho_1(u|v)$ is obtained by summing up the duration probabilities **above** u
- If the state was entered with density ψ over clocks, we can sum up $\rho_i(u|v)$ over $\pi(u)$ according to ψ to obtain the expected $\rho_i(u)$ as well as new densities ψ_i on clock values upon taking transition i



-

Definition

- For every **end** transition e_i outgoing from a state q with m active processes we define a density transformer $\mathcal{T}_{r_i}$
- The transformer $\mathcal{T}_{e_i}$ computes the clock density at the time when process i wins the race, given the density was ψ upon entering the state
- It is defined as $\psi_i = \mathcal{T}_{e_i}(\psi)$ if

$$\psi_i(x_1, \ldots, x_m, y_1, \ldots, y_m) =$$

$$\begin{cases} \displaystyle\int_{\tau > 0} \psi(x_1 - \tau, \ldots, x_m - \tau, y_1, \ldots, y_m) d\tau & \text{if } \begin{array}{l} x_i = y_i \wedge \\ \forall i' \neq i \; x_{i'} < y_{i'} \end{array} \\ 0 & \text{otherwise} \end{cases}$$

Concluding Remarks

- All those densities are piecewise polynomial and can be computed. The degrees of the polynomials and their piecewiseness grow with the number of steps
- So far we consider only acyclic DPA. For cyclic ones, we need some progress in fixpoint techniques for linear operators in the spirit of Asarin and Degorre (2009)
- A prototype implementation by M. Bozga, using a slightly different technique for computing volumes, can handle, for example a product of 2 SPDA with 10 steps each
- As in timed automata the larger is the ratio $(b-a)/a$ the more paths have to be considered
- There is still much to be done

Concluding Remarks

- All those densities are piecewise polynomial and can be computed. The degrees of the polynomials and their piecewiseness grow with the number of steps
- So far we consider only acyclic DPA. For cyclic ones, we need some progress in fixpoint techniques for linear operators in the spirit of Asarin and Degorre (2009)
- A prototype implementation by M. Bozga, using a slightly different technique for computing volumes, can handle, for example a product of 2 SPDA with 10 steps each
- As in timed automata the larger is the ratio $(b-a)/a$ the more paths have to be considered
- There is still much to be done
- Thank you