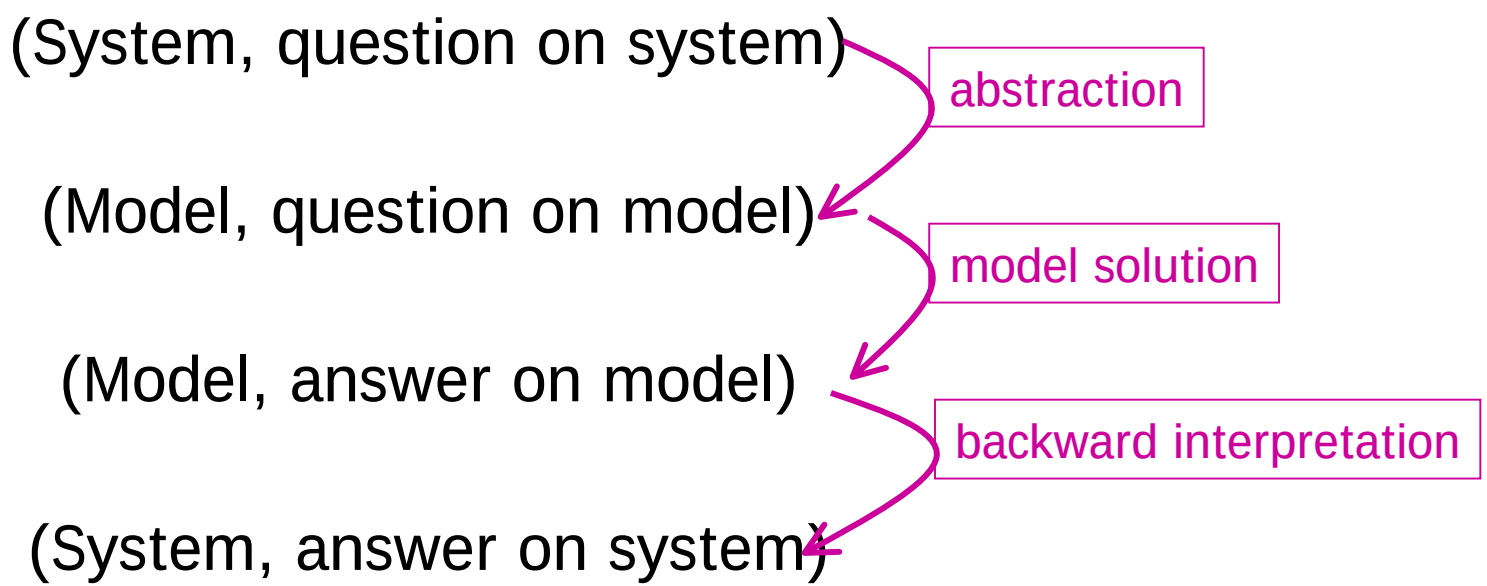
Systèmes temps réel probabilistes (model checking des chaînes de Markov à temps continu)

Prof.ssa Susanna Donatelli
Universita' di Torino, Italy
www.di.unito.it
susi@di.unito.it

Model checking of Continuous Time Markov Chain

Prof.ssa Susanna Donatelli
Universita' di Torino, Italy
www.di.unito.it
susi@di.unito.it

Context


(System, question on system)
abstraction
(Model, question on model)
model solution
(Model, answer on model)
backward interpretation
(System, answer on system)

Context

System type: discrete event systems

Categories of questions:

- qualitative -- will system reach a deadlock?
- quantitative -- will system reach a deadlock before time T?
- stochastic -- will system reach a deadlock before time T with probability >0.9 ?

Corresponding classes of models:

- finite automata (but also Petri Nets, Process Algebras, etc.)
- timed automata
- (continuous) time Markov chain (stochastic processes in general)

Context

Typical quesitons/properties

- qualitative -- reachability, deadlock, liveness, state/action condition, system evolution (path properties)
- quantitative -- timed reachability, timed system evolution (timed path properties)
- stochastic -- reachability in probability

Outline

- Verifying qualitative behaviour: CTL definition and model checking
- Verifying quantitative behaviour: CSL definition and model checking
- Beyond CSL, open problems and on-going work
- Bibliographical references

Verifying qualitative behaviour: CTL definition and model checking

Qualitative

- Simple qualitative model: automata (Kripke Structure)

$$M = (S, R, L)$$

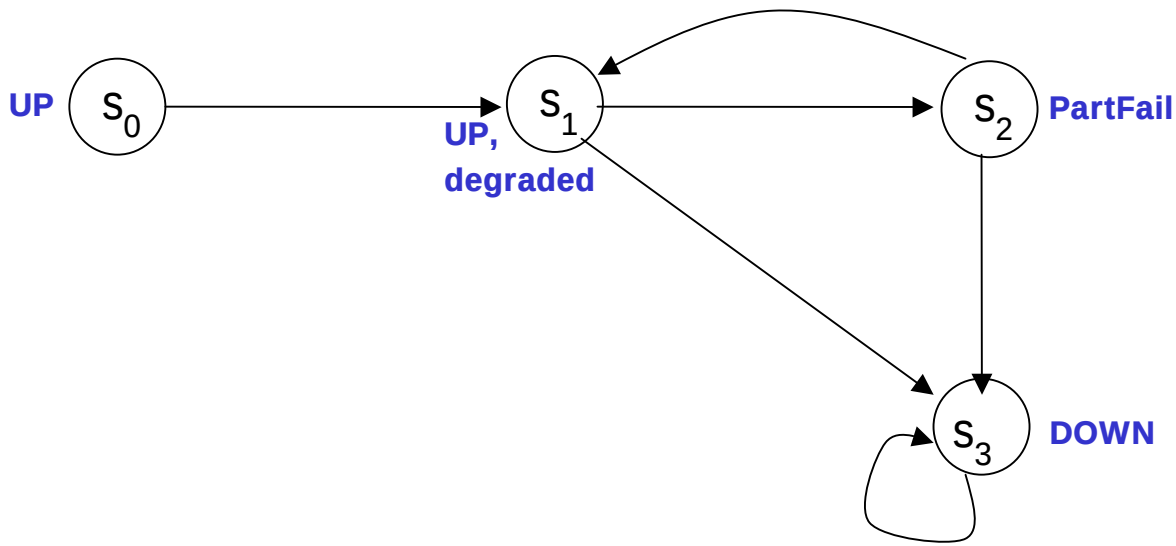
where

- S is a finite set of states
- R:S -->2^S, total function
- L: S --> 2^{AP}, with AP set of atomic propositions

- R(s) -- set of successors of s
- L(s) -- set of atomic propositions holding in s

Qualitative

Path of M = (S, R, L), starting in s : sequence of states for which $s_{(i+1)} \in R(s_{(i)})$ and $s_{(0)} = s$



Paths from $s_1 = \{s_1.s_2.s_1.s_2\ldots,\ s_1.s_2.s_3.s_3\ldots,\ \ldots.\}$

CTL main concepts

Computational Tree Logic, has been introduced by Clarke&Emerson in 1980

CTL allows to “speak about” states and paths

CTL is interpreted over Kripke structures

CTL has a *branching notion of time* (each event has many successors, at each time instant there are many possible futures)

CTL: Syntax

AP, set of atomic proposition. $p \in AP$.

CTL formulae:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid EX\varphi \mid E[\varphi U \varphi] \mid A[\varphi U \varphi]$$

E: “for some path”

A: “for all paths”

EX: “for some path next”

U: until

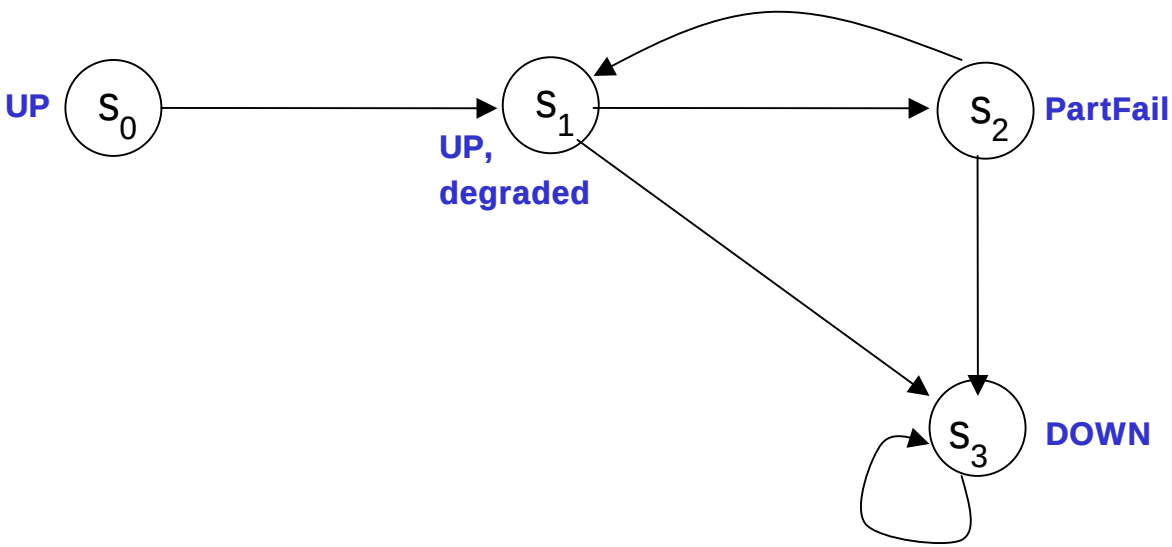
Note: syntactically correct formulas quantifiers and temporal operators are in strict alternation

Examples

Example of CTL formulae:

- E[UP *U* DOWN]
- A[UP *U* DOWN]
- EX (E[UP *U* DOWN])

which are the states that satisfy the formulae?



Model checking CTL

Problem definition: given a model M, a state s, and a CTL formula φ, does (M,s) |= φ?

In practice the model checking algorithms solve the problem: given a model M and a CTL formula φ, which are the states s, for which (M,s) |= φ?

As a by-product, at zero cost, the algorithm also computes all states that satisfy the subformulae of φ.

Model checking CTL

The algorithms starts with sub-formulae of length 1, and proceed by induction, until the formula of length $|\varphi|$ is computed

function Sat(φ: *CTL formula,* S: *set of State*): *set of* State

(* precondition: true*)

begin

if φ=true --> return S

[] φ=false --> return $\emptyset$

[] $\varphi \in$ AP --> return {s| $\varphi \in$ L(s)}

Model checking CTL

[] $\varphi = \neg\varphi_1$ --> return S - Sat(φ_1)

[] $\varphi=\varphi_1 \backslash/ \varphi_2$ --> return Sat(φ_1) $\cup$ Sat(φ_2)

[] $\varphi=EX\varphi_1$ --> return $\{s \in S | \exists (s,s') \in R \wedge s' \in Sat(\varphi_1)\}$

[] $\varphi=E[\varphi_1 U \varphi_2]$ --> return $Sat_{EU}(\varphi_1, \varphi_2)$

[] $\varphi=A[\varphi_1 U \varphi_2]$ --> return $Sat_{AU}(\varphi_1, \varphi_2)$

(* postcondition: $Sat(\varphi) = \{s \in S \mid (M,s) \models \varphi\}$

end

$Sat_{EU}(\varphi_1, \varphi_2)$ and $Sat_{AU}(\varphi_1, \varphi_2)$ are fixed point algorithms that use the axiom of the Until in terms of neXt and Until

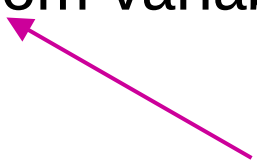
Verifying quantitative behaviour: CSL definition and model checking

Introducing time

Time was abstracted away

Taking time into account:

- duration of event is specified as an interval: in a single execution duration of event is non deterministically chosen from the interval
- duration of events are random variables: in a single execution (sample path) duration of event is drawn according to the random variable distribution



Basic model

Stochastic process: a family of random variables that model the evolution of the system under study

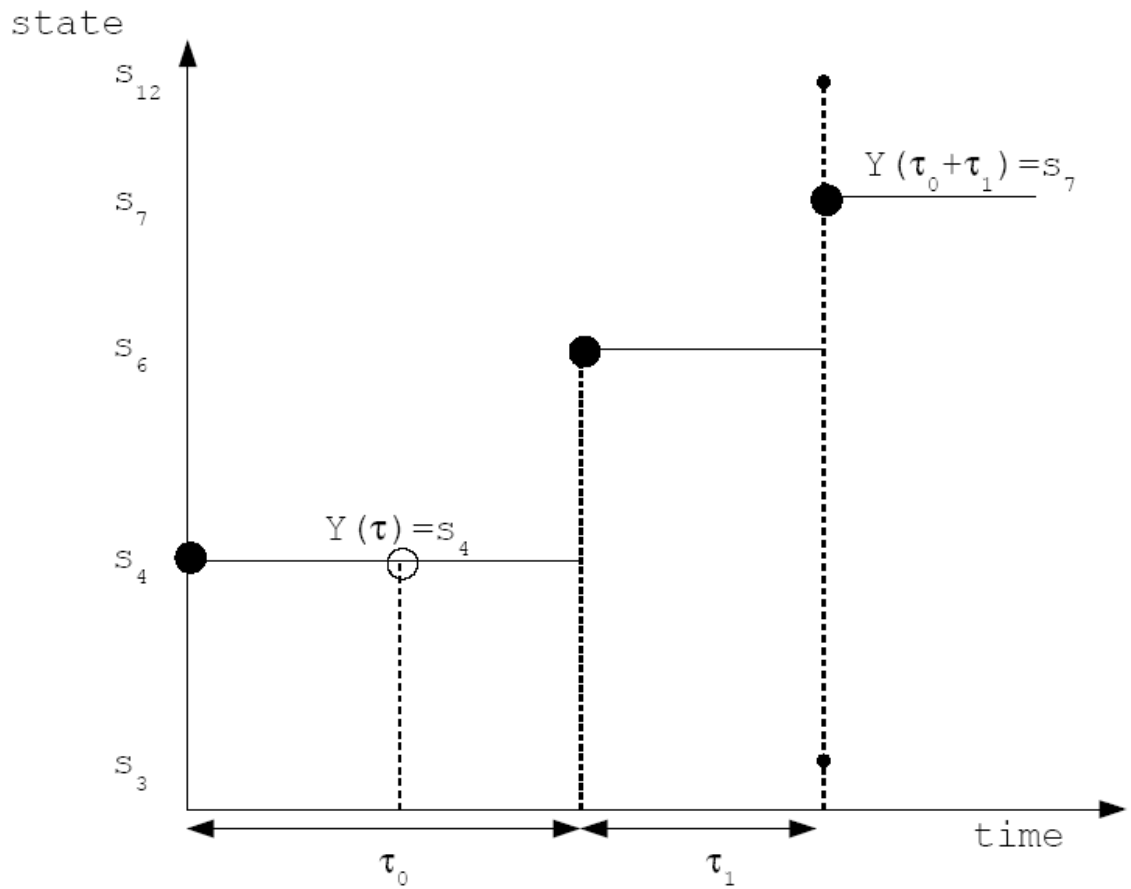
For the discrete event systems considered, we have two families of random variables:

- $\{X_i\}$ --- state of system after the i-th event
- $\{T_i\}$ --- time between i-th and (i+1)th event

Specifying a model requires the joint distribution:

$$\Pr(T_n < \tau \mid X_0 = s_{(0)}, \ldots\ldots, X_n = s_{(n)}, T_0 < \tau_0, \ldots\ldots, T_{n-1} < \tau_{n-1})$$

A sample execution



$T_0=\tau_0$

$T_1=\tau_1$

$T_2=0$

$T_3=0$

$X_0=s_4$

$X_1=s_6$

$X_2=s_3$

$X_3=s_{12}$

$X_4=s_7$

Discrete event stochastic process

Specifying a model requires the joint distribution:

$$Pr(T_n<\tau \mid X_0=s_{(0)}, \ldots\ldots, X_n=s_{(n)}, T_0<\tau_0, \ldots\ldots, T_{n-1}< \tau_{n-1})$$

Difficult to specify and to solve since the n-th state and n-th duration depend on the whole "history" of the system

..... simpler form of stochastic process

$$Pr(T_n<\tau \mid X_0=s_{(0)}, \ldots\ldots, X_n=s_{(n)}, T_0<\tau_0, \ldots\ldots, T_{n-1}< \tau_{n-1}) =$$

$$Pr(T_n<\tau \mid X_n=s_{(n)}) = 1 - e^{-\lambda_n \tau}$$

Exponential distribution

- Exponential distribution is memoryless
- Whenever the sojourn time in states is exponential, the stochastic process is a CTMC
- Exponential distribution is defined by its rate λ, and the mean value of an exponentially distributed rv is the inverse of the rate
- Exponentially distributed rv take values over all positive reals

CTMC

Labelled CTMC C = $(S, \mathbf{R}, \text{INIT}, L)$

- S, finite set of states
- the *rate matrix* $\mathbf{R} : S \times S \to \mathbb{R}_{\geq 0}$
- an *initial probability distribution* INIT on S,
- a *labelling function* $L : S \to 2^{AP}$.

Exit rate: $\mathrm{E}(s) = \sum_{s' \in S} \mathbf{R}(s, s')$

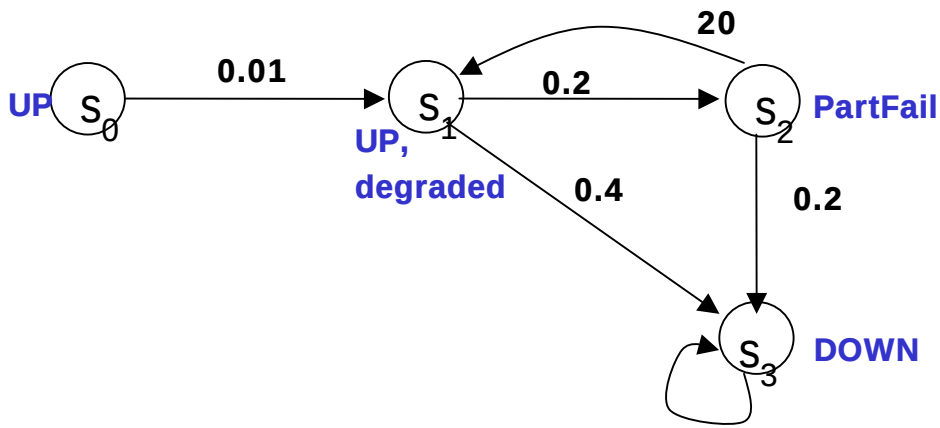
A state s is called *absorbing* if and only if $\mathrm{E}(s) = 0$.

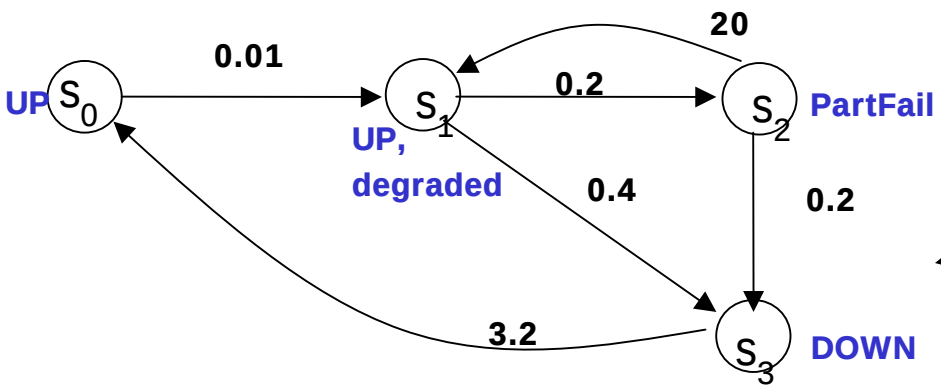
Path of a CTMC

$$\sigma = s_0 \xrightarrow{\delta_0} s_1 \xrightarrow{\delta_1} \cdots$$

CTMC

Two CTMCs





INIT = (0.9, 0.05, 0.05, 0)
E(s2) = 20.2

CTMC

Transient solution $\pi^C(INIT,s_j,t)$: probability of being at time t in state s_j , given the process starts with an initial distribution INIT

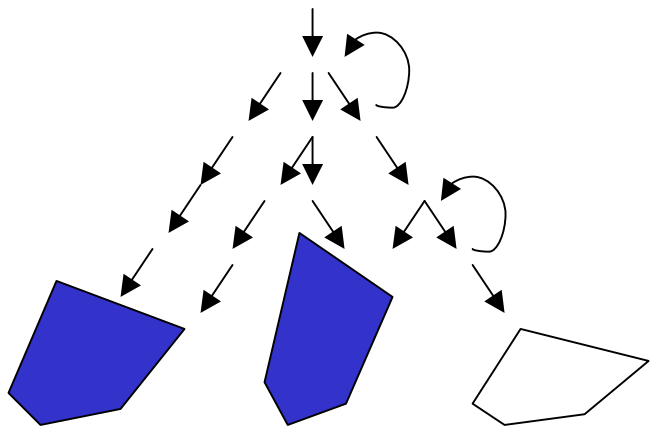
Transient solution $\pi^C(s_i,s_j,t)$: probability of being at time t in state s_j , given the process starts in state s_i

Steady state solution $\pi^C(INIT,s_j)$: probability of being in state s_j in the long run

If the CTMC is ergodic (graph is strongly connected) the initial distribution is irrelevant: $\pi^C(s_j)$

CTMC analysis

If the CTMS is not ergodic (the CTMC graph is not a single Strongly Connected Component) then the steady state may depend on the initial state, or on the initial state distribution



CTMC analysis

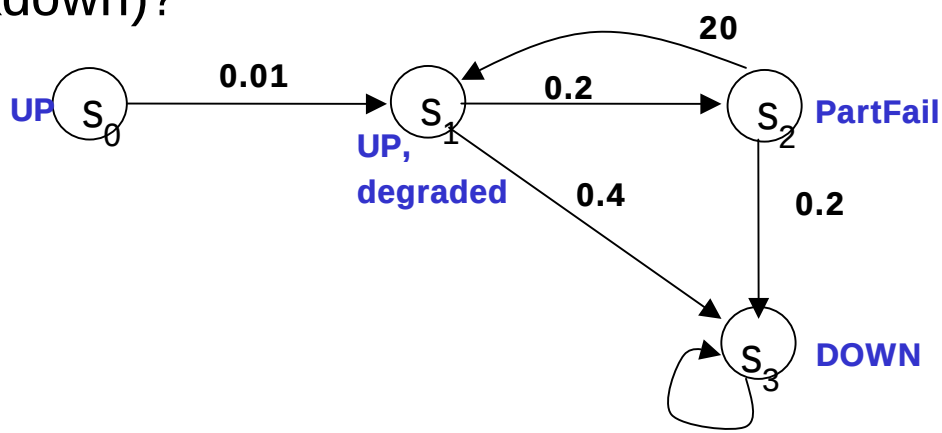
- Steady state solution π of an ergodic CTMC amounts to the solution of a set of linear equations $\pi.Q=0$ (usually solved iteratively)
- Steady state solution of a non ergodic CTMC requires first the computation of the probability of the bottom strongly connected components
- Transient analysis $\pi(t)$ usually performed using uniformization
- Performance indices defined on rewards
  - Associated to states
  - Associated to state transitions

Verifying behaviour

- Qualitative properties can be checked with CTL, forgetting the rate values

- Performance indices allow to draw probabilistic conclusion on the behaviour of the system, like probability of being in a UP state in steady state or at time t

- What about: probability of reaching a down state in within [2, 5.3] time unit, starting at 0 from state s_0 and never passing through a PartFail state (direct breakdown)?



Continuous Stochastic Logic

CSL syntax

$$\Phi ::= a \mid \Phi \wedge \Phi \mid \neg\Phi \mid \mathcal{P}_{\bowtie\rho}(X^I \Phi) \mid \mathcal{P}_{\bowtie\rho}(\Phi U^I \Phi) \mid \mathcal{S}_{\bowtie\rho}(\Phi)$$

$a \in AP$ is an atomic proposition,

$I \subseteq \mathbb{R}_{\geq 0}$ is a nonempty interval,

$\bowtie \in \{<, \leq, \geq, >\}$ is a comparison operator

$\rho \in [0, 1]$ is a probability

CSL syntax

$$\Phi ::= a \mid \Phi \wedge \Phi \mid \neg\Phi \mid \mathcal{P}_{\bowtie\rho}(X^I\Phi) \mid \mathcal{P}_{\bowtie\rho}(\Phi U^I \Phi) \mid \mathcal{S}_{\bowtie\rho}(\Phi)$$

- *State formulae* (atomic propositions and boolean expression) and *path formulae* (timed neXt and timed Until)
- $S_{<>\rho}(\Phi)$ is true in state *s* if the sum of the steady state probabilities of the Φ–states, computed using *s* as initial state, is <> ρ.
- $P_{<>\rho}(\Phi)$ is true in *s* if the probability of the paths leaving *s* which satisfy Φ is <>ρ.

CSL examples

Examples of CSL:

- $P_{\leq 0.01}(\text{true } U^{[10,20]} \text{ a})$
  - Satisfied in states from which the probability of reaching an a-labelled state after between 10 and 20 time units is no more than 0.01

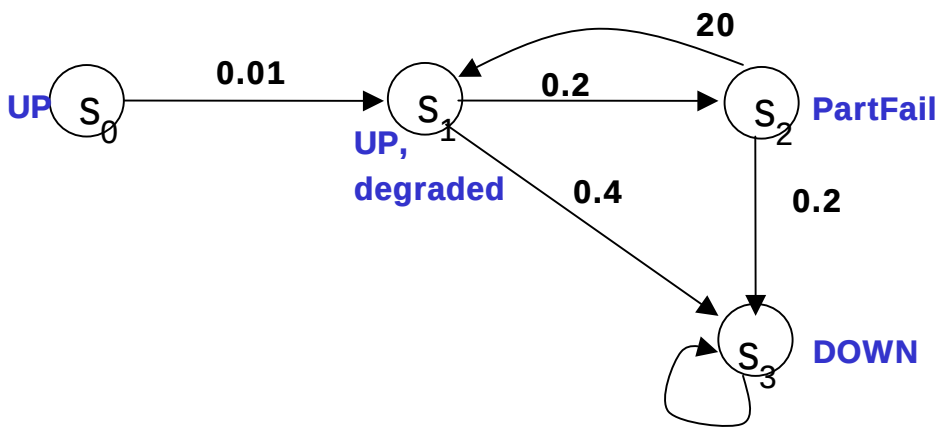
- $S_{>0.9}(a)$
  - Satisfied in states starting from which the probability of being in an a-labelled state in the long-run is greater than 0.9

- Nested formulae: e.g. $P_{\leq 0.1}(a\ U^{[10,20]}\ S_{>0.9}(b \wedge c))$

CSL examples

Probability of reaching a down state in within [2, 5.3] time unit, starting at 0 from state s_0 and never passing through a PartFail state (direct breakdown)?

$$s_0 \models P_{\leq 0.25}(\neg \text{PartFail } U^{[2,\ 5.3]}\ \text{DOWN})$$



CSL semantics

Labelled CTMC C = $(S, \mathbf{R}, \text{INIT}, L)$

$s \models$:

$$
\begin{array}{lll}
a & \text{iff} & a \in L(s) \\
\Phi_1 \wedge \Phi_2 & \text{iff} & s \models \Phi_1 \text{ and } s \models \Phi_2 \\
\neg\Phi & \text{iff} & s \not\models \Phi \\
\mathcal{S}_{\bowtie\rho}(\Phi) & \text{iff} & \pi^{C}(\alpha_s, Sat(\Phi)) \bowtie \rho \\
\mathcal{P}_{\bowtie\rho}(\varphi) & \text{iff} & Prob^{C}(s, \varphi) \bowtie \rho
\end{array}
$$

steady state of the φ-states in the CTMC, when it starts from s

Probability of the set of φ-paths of the CTMC that starts from *s*

CSL semantics

Remember, a path is $\sigma = s_0 \xrightarrow{\delta_0} s_1 \xrightarrow{\delta_1} \cdots$

A path satisfies a timed neXt or Until if:

$\sigma \models:$

$$X^I \Phi \quad \text{iff} \quad \sigma(1) \text{ is defined, and } \sigma(1) \models \Phi \wedge \Delta(\sigma, 0) \in I$$

$$\Phi_1 U^I \Phi_2 \quad \text{iff} \quad \exists t \in I . \sigma@t \models \Phi_2 \text{ and } \forall t' \in [0, t) . \sigma@t' \models \Phi_1$$

Accumulated time

state at time t in the sequence

Model checking

The model checking problem for CSL is: given a formula φ and a CTMC $\mathcal{M}$, that starts in state s, is it true that

$$(\mathcal{M},s) \models \varphi.$$

The algorithm works as for CTL, but now we need to compute steady state (easy) and probability of paths satisfying a path condition (not so easy)

Steady state formulae

If the CTMC C is ergodic

$$\pi^{C}(\alpha_s, S') = \sum_{s' \in S'} \pi^{C}(s'))$$

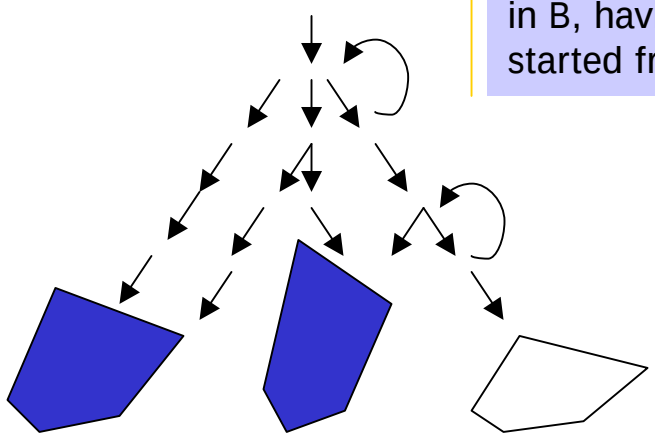
Steady state formulae

$$\pi^{C}(\alpha_s, S') = \sum_{B \in \mathcal{B}_C} \left(Prob(s, at_B) \cdot \sum_{s' \in B \cap S'} \pi^{B}(s') \right)$$

distribution over states reachable from s

prob. of being in B, having started from s'

prob. of being in state s", knowing you are in B



NeXt formulae

In a CTMC the events:

- next transition will occur in within *t*
- next state will be *s*

are independent

$$Prob^{C}(s, X^{I}\Phi) =$$

$$(e^{-\mathrm{E}(s)\cdot\inf I} - e^{-\mathrm{E}(s)\cdot\sup I}) \cdot \sum_{s' \in Sat(\Phi)} \mathbf{P}(s, s')$$

Probability of leaving the state in the interval I

Probability of going to a ϕ state

Until formulae

To compute the Until, we need to generate a CTMC $C[\phi]$ in which ϕ states are made absorbing

$$\mathbf{R}'(s,s') = \begin{cases} \mathbf{R}(s,s') & \text{if } s \not\models \Phi \\ 0 & \text{otherwise} \end{cases}$$

ϕ states are made absorbing

Path formulae: Until [0,t]

Probability of reaching a ψ state before time t, while passing only through ϕ-states

Track the behaviour of the process until either:

- a state fulfilling $\neg\phi$ or ψ is reached, or
- time *t* is reached

$$Prob^{\mathsf{C}}(s, \Phi U^{[0,t]} \Psi) = \sum_{s' \in Sat(\Psi)} \pi^{\mathsf{C}[\neg\Phi \vee \Psi]}(\alpha_s, s', t)$$

ψ states and $\neg\Phi$ states are made absorbing

Path formulae: Until [t,t]

Probability of being in a ψ state at time t, while passing only through ϕ-states between 0 and t: this is possible only in states in which both ψ and ϕ holds

$$Prob^{\mathsf{C}}(s, \Phi U^{[t,t]} \Psi) = \sum_{s' \in Sat(\Phi \wedge \Psi)} \pi^{\mathsf{C}[\neg\Phi]}(\alpha_s, s', t)$$

only $\neg\Phi$ states are made absorbing

Φ and ψ must hold together

Rationale: in a CTMC the probability of *entering* a state at exactly time *t* is zero

Open and closed intervals

Half-open and open intervals can be computed as for the closed case

Path formulae: Until [t,t']

Probability of reaching a ψ state in within t and t', while passing only through ϕ-states.

Paths must first reach t while staying on ϕ-states, and then we check the behaviour on the remaining time

$$Prob^{\mathsf{C}}(s, \Phi U^{[t,t']} \Psi) =$$

$$\sum_{s' \in Sat(\Phi)} \sum_{s'' \in Sat(\Psi)} \pi^{\mathsf{C}[\neg\Phi]}(\alpha_s, s', t) \cdot \pi^{\mathsf{C}[\neg\Phi \vee \Psi]}(\alpha_{s'}, s'', t' - t)$$

only $\neg\Phi$ states are made absorbing

Path formulae: Until [t,∞)

Probability of reaching a ψ state in within t and ∞, while passing only through ϕ-states.

Paths must first reach t while staying on ϕ-states, and then it is just probability of reaching a ϕ-states

$$Prob^{\mathsf{C}}(s, \Phi U^{[t,\infty)} \Psi) =$$

$$\sum_{s' \in Sat(\Phi)} Prob^{\mathsf{C}}(s, \Phi U^{[t,t]} at_{s'}) \cdot Prob^{\mathsf{C}}(s', \Phi U^{[0,\infty)} \Psi)$$

Tools

- PRISM: from the group of Martha Kwiatkowska at University of Birmingham, now in Oxford. Allows model checking of CTMC, DTMC and MDP. Input language is a guarded language

- ETMCC (now MRMC) from the group of Jost-Pieter Katoen et al. in Twente, Aachen, Munich. Input language are straight CTMC. Allows for rewards and paths specified also using actions

Beyond CSL

Problems, limits, on-going work

- State space explosion
- Steady state operator is a bit “odd” for performance people
- Most problems are more naturally sets in terms of “computing values” more than in terms of “yes/no answers”
- There is no way to account for an initial distribution different from all prob. accumulated in a single state
- Accumulating probability along paths was very often done in reliability studies by modifying the CTMC in an ad-hoc manner: the real innovation of CSL is that automates the process

Problems, limits, on-going work

- There are extension to account for the reward accumulated along paths
- Paths are defined only in terms of sequences of state properties - extension to actions have been defined
- CSL is for CTMC only, but there are extension to Semi-Markov processes (eCSL by Bradley et al. at Imperial College) and to CTMC with vanishing states (Cerotti et al. at Univeristy of Turin)

Problems, limits, on-going work

- There is no way of “concatenating” intervals

  $$P_{\leq 0.25}(\text{UP } U^{[2,4]} \text{ degraded } U^{[2,\ 5.3]} \text{ DOWN})$$

  is not a valid CSL statement

  A solution (CSL^{TA}) has been devised that allows the definition of the paths of interest using a Timed Automata (at QEST 2007 in Edinburgh, joint work Lamsade and University of Turin)

References

Path based performance measures

[1] G. Clark and J. Hillston. Towards automatic derivation of performance measures from PEPA models. In *Proceedings of the UK Performance Engineering Workshop*, 1996.

[2] W. D. Obal II and W. H. Sanders. State-space support for path-based reward variables. *Performance Evaluation*, 35(3-4):233–251, 1999.

References

Basic papers on CSL

[3] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton. Model-checking continuous time Markov chains. *ACM Transactions on Computational Logic*, 1(1):162–170, 2000.

[4] C. Baier, L. Cloth, B. Haverkort, M. Kuntz, and M. Siegle. Model checking action- and state-labelled Markov chains. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN'04)*, pages 701–710. IEEE Computer Society, 2004.

[5] C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen. On the logical characterisation of performability properties. In *Proceedings of the 12th International Colloquium on Automata, Languages and Programming (ICALP'00)*, volume 1853 of *LNCS*, pages 780–792. Springer, 2000.

[6] C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen. Model-checking algorithms for continuous-time Markov chains. *IEEE Transactions on Software Engineering*, 29(6):524–541, 2003.

References

[7] J. T. Bradley, N. J. Dingle, P. G. Harrison, and W. J. Knottenbelt. Performance queries on semi-Markov stochastic Petri nets with an extended continuous stochastic logic. In *Proceedings of the 10th International Workshop on Petri Nets and Performance Models (PNPM'03)*, pages 62–71. IEEE Computer Society, 2003.

[8] P. Buchholz, J.-P. Katoen, P. Kemper, and C. Tepper. Model-checking large structured Markov chains. *Journal of Logic and Algebraic Programming*, 56:69–96, 2003.

[9] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[10] H. Hermanns, J.-P. Katoen, J. Meyer-Kayser, and M. Siegle. A tool for model-checking Markov chains. *International Journal on Software Tools for Technology Transfer*, 4(2):153–172, 2003.

[11] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. In *Proceedings of the 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06)*, volume 3920 of *LNCS*, pages 441–444. Springer, 2006.

Self-references

J. Sproston and S. Donatelli: Backward Bisimulation in Markov Chain Model Checking , *IEEE Transactions on Software Engineering* 32(8), pp. 531-546, 2006.

S. Donatelli, S. Haddad and J. Sproston: CSL^{TA}: an Expressive Logic for Continuous-Time Markov Chain, In M. Harchol-Balter, M. Kwiatkowska and M. Telek, editors, *Proceedings of the 4th International Conference on Quantitative Evaluation of Systems (QEST'07)*, © IEEE-CS Press 2007. To appear.

Self-references

D. Cerotti, S. Donatelli, A. Horváth, J. Sproston: CSL Model Checking for Generalized Stochastic Petri Nets, In P. D'Argenio, A. Miner and G. Rubino, editors, *Proceedings of the 3rd International Conference on Quantitative Evaluation of Systems (QEST'06)*, pp. 199-210. ©IEEE-CS Press, 2006.

D. Cerotti, D. D'Aprile, S. Donatelli, J. Sproston: Verifying stochastic Well-Formed Nets with CSL Model checking tools, In K. Goossens and L. Petrucci, editors, *Proceedings of the 6th International Conference on Application of Concurrency to System Design (ACSD'06)*. ©IEEE-CS Press, 2006.

D. D'Aprile, S. Donatelli, J. Sproston: CSL Model Checking for the GreatSPN tool, In C. Aykanat, T. Dayar, I. Korpeoglu, editors, *Proceedings of the 19th International Symposium on Computer and Information Sciences (ISCIS'04)*, Lecture Notes in Computer Science 3280, pp 543-552. © Springer, 2004.