Probabilistic Methods in Concurrency

Lecture 6

Progress statements:
A tool for verification of probabilistic automata

Catuscia Palamidessi
catuscia@lix.polytechnique.fr
www.lix.polytechnique.fr/~catuscia

Page of the course:
www.lix.polytechnique.fr/~catuscia/teaching/Pisa/

Progress statements

- Progress statements
  - Proposed by Lynch and Segala
  - A formal method to analyse probabilistic algorithms

- **Definition (progress statements)**
  - Given sets of states S, T, and a class of adversaries A, we write

    S –A,p-> T

    if, under any adversary in A, from any state in S, we eventually reach a state in T with probability at least p
  - Furthermore, we write

    S unless T

    if, whenever from a state in S we do not reach a state in T, we remain in S (possibly in a different state of S)

Progress statements

- **Some useful properties**
  - If A is history-insensitive , S –A,p-> T, and T –A,q-> U, then

    $$S \text{ –}A\text{,pq-> } U$$

  - If S_1 –A,p_1-> T_1, and S_2 –A,p_2-> T_2 , then

    $$S_1 \cup S_2 \text{–}A\text{,p-> } T_1 \cup T_2$$

    where $p = \min\{p_1,p_2\}$
  - S –A,1-> S
  - If A is history-insensitive and S –A,p-> T and S unless T, and $p > 0$, then

    $$S \text{ –}A\text{,1-> } T$$

History insensitivity

- **Definition:** a class of adversaries *A* is history-insensitive if: for every $\alpha \in A$, and for every fragment of execution e, there exists $\alpha' \in A$ such that, for every fragment of execution e', $\alpha'(e') = \alpha(ee')$

- **Proposition:** The class of fair adversaries is history-insensitive

  Proof: Given α and e, define $\alpha'(e') = \alpha(ee')$. Clearly α' is still fair

Example of verification: the dining philosophers

- An example of verification using the progress statements.
- The example we consider is the randomized algorithm of Lehmann and Rabin for the dining philosophers
- We will show that under a fair adversary scheduler we have deadlock-freedom (and livelock-freedom), i.e. if a philosopher gets hungry, then with probability 1 some philosopher (not necessarily the same) will eventually eat.

The dining philosophers: the algorithm

State	action	description
R	think or get hungry	reminder region
F	flip	ready to toss
W	wait	waiting for first fork
S	second	checking second resource
D	drop	dropping first resource
P	eat	pre-critical region
C	exit	critical region
E_F	dropF	drop first fork
E_S	dropS	drop second fork
E_R	rem	move to reminder region

T

Example of verification: The dining philosophers

- Let us introduce the following global (sets of) states

  Try : at least one phil is in T={F,W,S,D,P}

  Eat : at least one phil is in C

  RT : at least one phil is in T, all the others are in T, R or E_R

  Flip : at least one phil is in F

  Pre : at least one phil is in P

  Good : at least one process is in a "good state", i.e. in {W,S} while his second fork f is not the first fork for the neighbor (i.e. the neighbor is not committed to f)

- We want to show that Try -A,1-> Eat for A = fair adv

Example of verification: The dining philosophers

- We can prove that, for the class of fair adversaries *A* (omitted in the following notation):
  - Try -1-> RT ∪ Eat
  - RT -1-> Flip ∪ Good ∪ Pre
  - Flip -1/2-> Good ∪ Pre
  - Good -1/4-> Pre
  - Pre -1-> Eat

- Using the properties of progress statements we derive

  Try -1/8-> Eat

- Since we also have Try unless Eat, we can conclude