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

    $$S \triangleleft A,p\rightarrow T$$

    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

    $$S \triangleleft$$

    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, $S \rightarrow_{A,p} T$, and $T \rightarrow_{A,q} U$, then
  $$S \rightarrow_{A,pq} U$$

- If $S_1 \rightarrow_{A,p_1} T_1$, and $S_2 \rightarrow_{A,p_2} T_2$, then
  $$S_1 \cup S_2 \rightarrow_{A,p} T_1 \cup T_2$$
  where $p = \min\{p_1, p_2\}$

- $S \rightarrow_{A,1} S$

- If $A$ is history-insensitive and $S \rightarrow_{A,p} T$ and $S$ unless $T$, and $p > 0$, then
  $$S \rightarrow_{A,1} T$$
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 $\alpha$ and $e$, define $\alpha'(e') = \alpha(ee')$. Clearly $\alpha'$ 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

<table>
<thead>
<tr>
<th>State</th>
<th>action</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>• R</td>
<td>think</td>
<td>reminder region</td>
</tr>
<tr>
<td></td>
<td>or</td>
<td>get hungry</td>
</tr>
<tr>
<td>• F</td>
<td>flip</td>
<td>ready to toss</td>
</tr>
<tr>
<td>• W</td>
<td>wait</td>
<td>waiting for first fork</td>
</tr>
<tr>
<td>• S</td>
<td>second</td>
<td>checking second resource</td>
</tr>
<tr>
<td>• D</td>
<td>drop</td>
<td>dropping first resource</td>
</tr>
<tr>
<td>• P</td>
<td>eat</td>
<td>pre-critical region</td>
</tr>
<tr>
<td>• C</td>
<td>exit</td>
<td>critical region</td>
</tr>
<tr>
<td>• E_F</td>
<td>dropF</td>
<td>drop first fork</td>
</tr>
<tr>
<td>• E_S</td>
<td>dropS</td>
<td>drop second fork</td>
</tr>
<tr>
<td>• E_R</td>
<td>rem</td>
<td>move to reminder region</td>
</tr>
</tbody>
</table>
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 $\text{Try} - A, 1 \rightarrow \text{Eat}$ for $A = \text{fair adv}$
Example of verification: The dining philosophers

- We can prove that, for the class of fair adversaries \( A \) (omitted in the following notation):
  - \( \text{Try} \rightarrow RT \cup \text{Eat} \)
  - \( RT \rightarrow \text{Flip} \cup \text{Good} \cup \text{Pre} \)
  - \( \text{Flip} \rightarrow \text{Good} \cup \text{Pre} \)
  - \( \text{Good} \rightarrow \text{Pre} \)
  - \( \text{Pre} \rightarrow \text{Eat} \)

- Using the properties of progress statements we derive
  \( \text{Try} \rightarrow \text{Eat} \)

- Since we also have \( \text{Try unless Eat} \), we can conclude