This pages contain the explanations for the five slides on proving time bounds for probabilistic systems. Each section describes a slide.

1 Probabilistic Time Bounds for Probabilistic Algorithms

We introduce a collection of new formal tools and methods for proving time bound properties for probabilistic systems. The motivating examples are the many randomized distributed algorithms that have been developed in the theory of distributed computing. The appropriate time bounds for such algorithms are generally *probabilistic time bounds*, i.e., assertions that some event occurs within a certain amount of time t, with at least a certain probability p.

It is notoriously difficult to obtain correct proofs for randomized distributed algorithms, and so it is worthwhile to develop systematic proof methods. We address two problems: define a model where time bound properties can be state formally (this answers the question "What does it mean?"), and develop proof methods for the new model (this answers the question "How is it proved?").

2 Theory

The theory we used is taken from [6, 3, 5].

Probabilistic timed automata. We modify the timed automaton model of [4] to incorporate probabilistic state transitions. Specifically, a probabilistic timed automaton (pta) is a labeled transition system (a state machine) whose transitions (steps) are of the form (s, π, Π), where Π is a probability distribution whose domain is a subset of the states of the algorithm. The meaning is that when the step occurs, a probabilistic choice according to distribution Π is used to determine the new state. A probabilistic timed automaton (pta) has a combination of nondeterministic and probabilistic branching in its timed executions, where nondeterminism is involved in the choice of the next step.

Meaningful probabilistic statements about the behavior of a pta must be based on a probability distribution on its timed executions. In order to obtain such a distribution, we must resolve all nondeterministic choices, making all the branching probabilistic. We do this by hypothesizing an entity called an *adversary* that determines the next step; formally, an adversary is a function that, given the finite execution performed so far, returns the following (possibly probabilistic) step. A fixed adversary $\mathcal{A}$ gives rise to an execution tree with only probabilistic branch points, and thereby to a probability distribution on timed executions.

An adversary is said to be *admissible* provided that all the timed executions that appear in its distribution are admissible, i.e., provided that it always allows time to pass to ∞. When we claim that an algorithm (pta) guarantees some property P with at least probability p, we mean that for every admissible adversary $\mathcal{A}$, the probability of P in the probability distribution of timed executions generated by $\mathcal{A}$ is at least p.

3 Progress Rules

One practical method for proving probabalistic time bound properties is based on formulating Markov-style statements of the form $\mathcal{U} \xrightarrow[p]{t} \mathcal{U}'$, where $\mathcal{U}$ and $\mathcal{U}'$ are sets of states. This statement means that for every admissible adversary $\mathcal{A}$, if the algorithm is started in a state in $\mathcal{U}$, then in the probability distribution generated by $\mathcal{A}$, the probability that a state in $\mathcal{U}'$ is reached within time t is at least p. Such statements can be combined:

Lemma 3.1 *If* $\mathcal{U} \xrightarrow[p]{t} \mathcal{U}'$ *and* $\mathcal{U}' \xrightarrow[p']{t'} \mathcal{U}''$ *then* $\mathcal{U} \xrightarrow[pp']{t+t'} \mathcal{U}''$.

4 Coin Lemmas

In most situations where a property of the form $\mathcal{U} \xrightarrow[p]{t} \mathcal{U}'$ is true, the probability p arises as the probability that certain designated random choices are made in a favorable way. In such a case, the statement can usually be proved by fixing the results of those choices in the favorable way and converting all the remaining random choices to nondeterministic choices. This gives rise to an ordinary (non-probabilistic) timed automaton A. Proving that the desired time bound holds *with certainty* for A implies that it holds with probability p in the original pta. The soundness of this strategy is justified by a series of *coin lemmas* in [5].

5 Advantages

In the new framework the proofs of properties for probabilistic algorithms are based on rigorous arguments, and thus are more reliable than proofs based on informal arguments.

Surprisingly, the formal proofs are easier to understand than the informal proofs, since in our framework probability is used only where necessary. We have so far carried out two significant proofs of probabilistic time bound properties for randomized distributed algorithms. In [3] the Randomized Dining Philosophers algorithm of Lehmann and Rabin is shown to guarantee progress in constant expected time. A larger and more complex example is the Aggarwal-Kutten randomized algorithm for finding a spanning tree in a network based on an unknown undirected graph G [2]. This algorithm is a randomized distributed algorithm of typical difficulty; the state of the art in the distributed algorithms area has till now not permitted careful proofs of such algorithms. Nevertheless, in [1] it is proven that a spanning tree is found in expected time in the order of the diameter of G using progress rule and coin lemma methods. There are two interesting points to be noted about the proof. First, as is common when careful proofs are carried out for complicated algorithms, a mistake was found in the original code. (It was easy to fix.) Second, although the final proof is fairly lengthy, the parts of it that deal with probabilities constitute only a few pages. One effect of our work on systematizing such proofs has been the reduction of reasoning about probabilistic systems to reasoning about non-probabilistic systems.

Finally, since the new probabilistic model is a generalization of the ordinary labeled transition system model, the existing verification techniques carry over. In particular, we are currently extending the verification method based on simulation relations. Some preliminary work is in [6]; the full work will appear in [5].

References

[1] S. Aggarwal. Time optimal self-stabilizing spanning tree algorithms. Master's thesis, Massachusetts Institute of Technology, Laboratory for Computer Science, 1994.

[2] Sudhanshu Aggarwal and Shay Kutten. Time optimal self stabilizing spanning tree algorithms. In R.K. Shyamasundar, editor, *13th International Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 761 of *Lecture Notes in Computer Science*, pages 400–410, Bombay, India., December 1993. Springer-Verlag.

[3] N. Lynch, I. Saias, and R. Segala. Proving time bounds for randomized distributed algorithms. In *Proceedings of the* 13^{th} *Annual ACM Symposium on Principles of Distributed Computing,* Los Angeles, CA, pages 314–323, 1994.

[4] N.A. Lynch and F.W. Vaandrager. Forward and backward simulations – part II: Timing-based systems. Technical Report MIT/LCS/TM-487, MIT Laboratory for Computer Science, September 1993.

[5] R. Segala. PhD thesis, MIT, Dept. of Electrical Engineering and Computer Science, 1994. In progress.

[6] R. Segala and N. Lynch. Probabilistic simulations for probabilistic processes. In B. Jonsson and J. Parrow, editors, *Proceedings CONCUR 94,* Uppsala, Sweden, volume 836 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.