----------------------- REVIEW 1 ---------------------  
SUBMISSION: 7  
TITLE: A Probabilistic Choreography Language for PRISM  
AUTHORS: Marco Carbone and Adele Veschetti  
  
----------- Overall evaluation -----------  
SCORE: 2 (accept)  
----- TEXT:  
This paper presents a probabilistic choreography language for the probabilistic model checker PRISM.   
  
The paper embraces choreographic programming as an alternative to process-oriented programming to prevent undesired emergent behaviours. As usual in such approaches, the idea is to sacrifice flexibility in favour of predictability, by imposing specific specific structured patterns in the communication. The introduction should be more clear about which kind of concurrent process is amenable for choreographies, and for which class of concurrent processes choreographies are not suitable. The authors could easily anticipate some of the ideas on this matter from section 7 (paragraph "Discussion and Future Work").  
  
The introduction includes a motivating example from the PRISM model set. The idea of the authors is that the PRISM model is hard to understand, while the same example described in the proposed choreographic language is easy to undersand. I am afraid that this passage needs more effort to work as intentended. First of all, it does not help that there is no explaination about what the model is about. There is a user and a checkout. What are they? What are they trying to achieve? Second, is the choreography not just the product of both modules (seen as finite state machines)? Would that be the general idea, i.e. to model the state space as one finite-state machine directly?  
  
The above question seems to be relate to the conjecture in Section 7 about the performance of generating the CTMC/DTMC models direactly instead of via projection. More words on this in the introduction could help appreciating the choreographic approach in the setting of PRISM.  
  
I also suggest clarifying a bit better the contributions. In item (2), I would clearly state that/if the formal semantics of PRISM is faithul to the original one (I assume so) and that the purpose is to present the minimal fragment of PRISM that is needed for PRISM code generation from choreographies. In the description of the last item one should say that it implements the compiler of the third item. The current phrasing seems to suggest that two different compilers have been defined/implemented.  
  
A nice aspect of the paper is indeed that the approach has been fully formally defined and prove correct (at a high level of detail) and implemented into a working tool  
  
The goal of the experiments is however not very clear. The authors' claims are, for example:  
  
"using choreographies is beneficial in the presence of concurrency"  
"facilitating powerful analysis while maintaining expressive clarity.  
  
are the experiments aimed at backing up those claims?  
  
The authors use 3 examples from the PRISM benchmarks. They have modelled them in PRISM. There is however no discussion or experiment aiming at assessing "expressive clarity". Actually, one example is a sequential system (where the use of choreographies seems unnatural), and for another one the authors explains that "The PRISM model we created is more verbose than the one in [6]". I recommmend the authors to provide more convincing examples where the choreographic model is, at least, shorter in terms of lines of code.   
  
The authors do not test if the generated models are semantically equivalent (e.g. bisimilar) to the original ones. Instead they compare specific properties of interest of the models (e.g. some probability reachability property). These experiments are not explained in enough detail (e.g. what is "T" in the plots? Are you considering some bounded reachabiity property? What is the exact property being evaluated?). It is also not clear why simulations are used to estimate probabilities, given that PRISM supports exact probability calculations.

-> l’avevamo rimosso per bisogno di spazio, aggiungere di nuovo?  
  
Overall, this is an original piece of work, presenting a good idea of defining a probabilistic choreography model well integrated with a state-of-the-art probabilistic model checker. I support the idea but a scepctic reader would need more convincing examples and experiments to appreciate the alternative modelling approch. This should be easy to achieve in a final version of the paper.  
  
  
  
----------------------- REVIEW 2 ---------------------  
SUBMISSION: 7  
TITLE: A Probabilistic Choreography Language for PRISM  
AUTHORS: Marco Carbone and Adele Veschetti  
  
----------- Overall evaluation -----------  
SCORE: 3 (strong accept)  
----- TEXT:  
This paper was well structured and well-written. The core contribution was a probabilistic choreography language with a defined translation to PRISM. The translation was then implemented and benchmarked.   
  
Theory:  
The projection section defined a translation and a correctness criterion (a variant of end point projection) with a proof sketch that made sense to me.  
  
Implementation:  
The implementation section was thorough and gave me an idea of how the choreographies were actually compiled into PRISM. The benchmarking section exceeded my expectations and was well-explained.   
  
Examples:   
The choreography section had two examples in it which were helpful. The PRISM section had one example in it but I got confused trying to figure out if it was connected to the first two examples. The projection section didn't have any examples in it and it would have benefited from one. In general, this paper would have been even stronger if there was a running example throughout the paper -- perhaps a particular probabilistic choreographic program that was then projected to PRISM and implemented (and maybe even benchmarked?).

-> ho aggiunto esempio in sezione prism

Related work:  
While there does seem to be a dearth of work in the choreographies/session types on probabilistic programming, I am wondering if there is any other work in the distributed systems world that handles probabilistic programming, even if the theoretical model (or lack thereof) is very different. Unfortunately, I do not have any specific references to suggest, but it might be interesting to situate this work more broadly, not just in the PL + distributed computation space.   
  
Also, since your work makes use of state machines, I am wondering if there are connections CFSMs of something similar (since multiparty session types can be modeled as communicating FSMs, perhaps choreographies are similar)?  
  
  
  
----------------------- REVIEW 3 ---------------------  
SUBMISSION: 7  
TITLE: A Probabilistic Choreography Language for PRISM  
AUTHORS: Marco Carbone and Adele Veschetti  
  
----------- Overall evaluation -----------  
SCORE: 2 (accept)  
----- TEXT:  
This paper introduces a novel (pure) probabilistic choreography language designed specifically for PRISM processes. It extends a standard choreography language by allowing the continuation of each interaction prefix to be chosen probabilistically from a set of available continuations. Additionally, the paper demonstrates how probabilistic choreographies can be implemented as PRISM processes, providing a projection function. The correctness of this projection is established by showing that processes obtained through projection faithfully reflect and preserve the reduction semantics of choreographies.  
  
Furthermore, the paper discusses the implementation of this approach. A Java application is used to parse choreography definitions and generate PRISM code. The paper illustrates the application of the proposed choreography language through three case studies: Dice, Proof of Work Bitcoin Protocol, and Hybrid Casper Protocol. The experiments show that these protocols can be defined more concisely using the proposed language, and that the resulting PRISM code serves the same purpose as existing implementations of the examples.  
  
Overall, the paper is well-written and presents an interesting contribution by providing a higher-level language for obtaining PRISM programs. Although there are a few points (listed below) that could benefit from clarification, I'm in favour of acceptance.  
  
- The projection operation assumes all non-participants in an interaction behave the same in all branches. I wonder if this restriction is due to technical simplifications, intrinsic problems when combining probabilities with lack of knowledge of choices. Further analysis would be beneficial.  
  
- The paper does nor require the usual condition of non-self-interaction (as evident in Listing 1.4). Additionally, clarification is needed on how projection works in cases where non-self-interaction is not enforced. Are the projection rules ordered?  
  
- The examples provided in the paper do not utilise generalised interactions with possible many receivers. It would be worth discussing whether this feature is necessary and, if so, including it in the examples.

--> abbiamo questa feature, ma non abbiamo esempi   
  
- In Listings 1.5 and 1.6, indexes "i" are used and the description mentions "the code depicts miners." Clarification is needed on the interpretation of a prefix like Miner[i], which may represent multiple miners or a single miner with a specific index (I would expect the later because the formalisation does not handle several senders)

🡪 viene spiegato a pag 11  
  
Minor comments  
  
page 4:  
  
- "One of the i branches is selected" -> j ∈ J instead of i?  
- The definition of the update operator is unclear. It would be helpful to explain the meanings of "u" and "&". I assume is ordered substitution.