[First of all, we thank the reviewers for their insightful comments.]

Reviewer #1: In this paper open parameterized networks of synchronized automata (pNets) are used to define and reason about incompletely defined systems. In this respect, open pNets are good candidates for specifying and reasoning about BIP architectures that comprise a set of coordinator components and a set of connectors, a subset of which is dangling, i.e., the system is not completely defined.

I find the contribution of the paper interesting and clear. In particular, it consists of 1) an open automaton generation algorithm 2) a strategy for checking satisfiability of open transitions (using the Z3 solver) and 3) the application of the approach to some (relatively small) case studies.

However, some parts of the paper were not clear to me:

1) In page 4, the authors argue that "the interaction feature in architectures does not handle data-sensitive interaction constraints. Using encoding of architectures into open pNets was an interesting alternative to a direct extension of architecture semantics". My understanding is that the direct extension of architecture semantics should be straightforward in order to handle data sensitive interaction constraints (e.g., the approach could follow the BIP data transfer mechanisms). I would like to understand what drove the authors to this alternative extension and why they consider it interesting in comparison to the direct extension of architecture semantics.
[Thank you for pointing this out. The paragraph referred to here has been included in this text by mistake. We have now removed it, adding a sentence to explain that this paper provides verification tools for individual BIP architectures, which were out of the scope of the cited previous work.]

2) The preliminaries section is not very clear. I would advise the authors to move the examples presented in Figures 1 and 2 in the beginning of Section 2 (before Definitions 1,

2, and 3) and explain first all the elements of the figures in detail and then make the connection with the formal definitions.

[We have restructured the running example presentation as suggested, introducing the example just after Definition 1, giving their algebra presentation. Later we use the same example (and have enhanced their explanations) for each of the definitions of PLTS, pNet, OT, etc.]

3) In section 4.3, I wonder whether Z3 returned 'unknown' for any of the cases. If yes, I would be interested in understanding what triggered the 'unknown' response. In general, it would be interesting if the authors could check and comment on the limits of the proposed approach.

[Yes, Z3 can return ‘unknown’, as there can be of course some intrinsically undecidable theories, but also because user defined data domains may be incompletely axiomatized… We have commented on this in Section 4.2]

4) The (variable) names in Figure 4 are different from the ones used in Section 2.2, which is confusing.

[We have adjusted the variable names in Figure 4 according to Section 2.2, similarly for Figure 5 (OT5).]

5) The initial transition of the second open automaton in Figure 1 has been removed from the version shown in Figure 4.

[We now explain at the beginning of the OTs that the initial transition is treated as an initialization of the state variables (“v:=0”).]

6) It is not clear why the authors refined Figure 6 (what are the differences from the original version used in [33]?).

[We have added footnote 2 explaining the reasons and briefly outlining the difference.]

7) I am not convinced that the approach scales, as the authors claim in the conclusion section. The presented examples are relative small.
[We have clarified our discussion of scalability in the conclusion.]

I also have the following questions:

Can the composition of architectures shown in Figure 6 be specified through an architecture style with n T components and n C components? Could the described approach be used at the level of architecture styles and not of architectures?
[Indeed. We have added a discussion in the conclusion section]

Typos and minor comments:
- Page 8: controler
- Page 8: in example 1 can you add a reference to Figure 2 (for the pNet EnableStateCompLeft)?
- Page 10: futher
- Page 17: tracability, relevent
- Page 20: syncrhonisations
[DONE]

Reviewer #3: The paper focusses on the so-called pNets, i.e., parameterized networks of synchronized automata, a formalism introduced to provide a specification of distributed (synchronous or asynchronous) systems. More precisely, an open variant of pNets is considered, able to model open systems by allowing for the presence of "holes", subsystems whose behavior is only partially specified (e.g., by indicating only the actions they can possibly execute). In some earlier work, (some of) the authors proposed an operational semantics for pNets in term of open automata: states are characterized by a

set of state variables and transitions between states use conditions and expressions involving the behavior of the holes.

In the present paper, the authors propose an algorithm for computing such semantics (under some finiteness conditions). In order to prune transitions that can only occur under some unsatisfiable condition (and thus which have no ground instance), the authors show how satisfiability of the transition predicates can be checked with the SMT solver Z3. An improved version of the algorithm, which tries to generate only the reachable part of the automaton is also proposed, referred to as the "smart" algorithm. This is not straightforward, since states and transition are of symbolic nature. The algorithms are implemented in a prototype tool.

The approach is illustrated on two examples: a simple one, based on the encoding of the enable operators of Lotos, and a larger one inspired to an industrial case study.

* Evaluation *

The problem faced in the paper is definitively of interest: generate a symbolic description of the semantics of an open distributed system that can be possibly used to verify property of the system or to show its equivalence with some abstract specification. The theoretical contribution looks somehow thin (an improvement of an existing algorithm and the integration with an SMT solver). Still, from a practical point of view, an improved algorithm and its concrete implementation can be important. Moreover, developing the framework at this level of generality (generic data types, operators, synchronization model, etc.) requires to take care of many technical details.

In my opinion, however, the organization of the material and, more generally, the presentation are not very satisfactory. There are a lot of technical definitions that for a reader unfamiliar with pNets (like I am) can be difficult to digest without a proper guide.

Each definition should be motivated, and, possibly, illustrated on the simple running example (the one inspired to Lotos). There are several typos, small mistakes, sloppy sentences that make some parts of the paper difficult to follow (see the minor comments). The explanation of the way some implementation problems have been faced is often too abstract to be appreciated.

[We have reorganized the first sections, inserting our running example after each main definition. We hope it significantly helps to follow the presentation. We have corrected many typos/sentences of course.]

I think that the paper needs a major revision before being considered for publication.

* minor comments *

page 2, line 41:

This item is difficult to follow, please rephrase

[Rephrased.]

page 3, line 35

capacity?

[Corrected: capability.]

page 4, line 14

dependant?

[Corrected: dependent.]

page 5, line 17

E_V \cap A_V = \emptyset: why?

[Concerning the last change in the Term algebra section, we decided that $A_V \cap E_V = \emptyset$ is useless and we indeed use actions as IDs in parameterized actions. So we remove the constraint, and add a short explanation.]

page 5, line 46

Please, explain the role of the FUN constructor via an example

[As Examples 1 has been pulled after Definition 1, we now add an example of the FUN in Example 1.]

page 6,

line 36: what is I_V?

line 37: what is g_k? a condition?

[Explained:

I_V is the set of indexed sets with a range depending on variables of V, and g_k is already explained in the next several sentences.]

page 7, line 47:

composition transition -> composed transition?

[Corrected.]

page 8, line 13 (Def. 5)

What is the role on I, I'?

[We have removed I, I' because of the absence of the definition of LTS_i^i\in I in the definition of open automaton.]

page 8, line 34:

why not asking directly: SV_k = (a_i)^{i \in I}, (b_j)^{j \in j}, v ?

[Corrected.]

page 8, line 41

forall z -> forall y1?

[Corrected.]

page 10, line 15

"We prune UNSAT transitions": is this always decidable? Which hypotheses do you need on the theory you are working with?

[The answer is similar to the comment 3) of Reviewer #1, Z3 will return ‘unknown’, if there is some intrinsically undecidable theories, or user defined data domains are incompletely axiomatized. We have discussed how to deal with it in the following section.]

page 10, line 50:

futher

[Corrected.]

page 11:

I got confused when after filtering I found section 4.2, which, as far as I can see, is discussing the same issue. Can this be better organised?

[We have shortened the paragraph on filtering. The filtering here just eliminates the transitions of the first case.]

page 12, line 30

It took me some time to understand what unreachable transitions are. Again, some more explanation and/or an example would be helpful.

[We have added the reason to explain the existence of unreachable transitions.]

page 12, line 47

"From a practical point of view, very little can be gained ..." why?

[The reason is that it costs time to traverse the set of OTs to find the unreachable OTs which are usually a small part of the OTs. We added a short explanation.]

page 17, line 36

tracability

[Corrected.]

page 19, line 50

"they are bisimilar": unclear to me what is bisimilar to what and according to which notion of bisimulation.

[We have explained that the open automata are equivalent with respect to a notion of (symbolic) FH-bisimulation we defined in previous work.]

page 21, properties (1) and (2)

Are these properties relevant for the exposition? I mean, can they be enforced/checked in the proposed framework?

[These properties are very important for the further work about analysis (model-checking, equivalence checking) that will be done on the generated open automaton. We have added this explanation.]