We would like to thank all the reviewers for their insightful and constructive feedback. We are glad that they liked our framework of using natural language for planning, our environment, and our large-scale dataset.

We first address several common points:

- Reviewers were curious if we can sample novel instructions autoregressively. The RNN-Generative produces well-formed language and we can indeed use the generated instructions, instead of the pre-selected top 500 instructions, to instruct the executor. This model can get comparable win rate to the RNN-Discriminative in Table3. We will include this number and samples of generated instructions in the camera ready.

- We also want to re-emphasize evidence for the importance of the compositionality of natural language. We show this by comparing RNN/BoW models (compositional) against OneHot (non-compositional). Further, we showed that it is important to consider a sequence of history instructions in such complex context around line250. This result shows the need to compose information from across multiple instructions for good performance.

Finally, we appreciate the reviewers for suggesting additional citations and interesting future directions. We will add those in the camera ready.

Response to Reviewer 1

Natural language has several advantages over latent programs. Firstly, natural language is highly expressive and can be applied to many domains where actions would be difficult to represent with programs. At the least, the space of programs would likely have to be engineered for each new domain, which is not the case with natural language. Secondly, gathering supervision for natural language actions is possible with the framework we introduce.

We certainly do not claim to be "solving this task" in the paper. In Table3, the comparison is made between a hierarchical agent that uses language and an agent that does not use language. Both agents are trained on the same dataset. One of our major claims is that having such hierarchy with natural language as intermediate instructions is helpful. Training an RL agent for such RTS environment is feasible, as demonstrated by the DeepMind's effort in Starcraft II, but remains challenging and highly computationally expensive.

Many simple instructions such as "attack", and "build peasants" are very frequent, and can be used in many situations. Please see Table7 in appendix for most frequent instructions with their frequency.

We have indeed evaluated the agents against rule-based bots and the differences between different models and overall trend is similar to the results in Table3. Training with selfplay with unit-level control is challenging and beyond the scope of this paper.

We generate actions for all units at once, ignoring their orders and dependency.

Response to Reviewer 2

Thanks for the terminology suggestion, and the missing reference. At test time, the language is clearly latent, because it is intrinsic to the model's decision making process and has no other effects. However, at training time we rely on the supervised data to learn to use natural language. We agree that the distinction could be clearer, and will update the paper.

We have included description of the rule based bots used for collecting data in the appendix due to page limit. Please note that we do not compare our trained models against rule-based bots but rather compare models that uses language against a baseline that does not. Therefore the details on those bots are less important. The RNN used in the paper is a one layer LSTM.

Response to Reviewer 3

The claim for compositionality is mainly demonstrated in OneHot model (non-compositional) vs BoW/RNN models (compositional). As we can see from the both Table2 and Table3 that the compositional models dramatically outperform the non-compositional model in terms of both likelihood and win rate. In addition, although the RNN executor and BoW executor has little difference in terms of likelihood, the RNN instructor outperforms BoW instructor with a relative large margin in terms of both likelihood and win rate.

We can play the game by typing the instruction to instruct the executor. The executor responds accurately to those instructions. We can also generate instructions with trained instructor and control units ourselves through game interface.

The baseline model is trained with supervised learning while other more complex RTS agents are trained with reinforcement learning with significantly more computation resources and samples.

We believe that our method that factorizes unit actions to type and argument classifiers is more generalizable and scalable. A similar approach was also adopted by OpenAI's Dota bot trained in large scale RL setting.