Can We Teach a Transformer To Reason About Effects of Actions?

Pratyay Banerjee, Chitta Baral, Man Luo, Arindam Mitra[2], Kuntal Pal,
Tran C. Son[3], Neeraj Varshney,
Arizona State University, Microsoft[2], New Mexico State University[3]
pbanerj6,chitta,mluo26,kkpal, nvarshn2@asu.edu,
arindam.mitra@microsoft.com, stran@nmsu.edu

Abstract

A recent work has shown that transformers are able to "reason" with facts and rules in a limited setting where the rules are natural language expressions of conjunctions of conditions implying a conclusion. Since this suggests that transformers may be used for reasoning with knowledge given in natural language, we do a rigorous evaluation of this with respect to a common form of knowledge and its corresponding reasoning – the reasoning about effects of actions. Reasoning about action and change has been a top focus in the knowledge representation subfield of AI from the early days of AI and more recently it has been a highlight aspect in common sense question answering. We consider four action domains (Blocks World, Logistics, Dock-Worker-Robots and a Generic Domain) in natural language and create QA datasets that involve reasoning about the effects of actions in these domains. We investigate the ability of transformers to (a) learn to reason in these domains and (b) transfer that learning from the generic domains to the other domains.

1 Introduction

Giving the long pursued goal of AI - starting with McCarthy's Advice Taker (McCarthy 1959) - of having systems that can reason with explicitly given general knowledge as a motivation, the recent work (Clark, Tafjord, and Richardson 2020a) studies the ability of transformers to "reason" with facts and rules given as natural language sentences. They show that in their limited setting where rules are natural language expressions of conjunctive implications of the form "[$\wedge$ condition]* $\rightarrow$ conclusion" transformers can "*reason*" with such rules (and facts) given in natural language and answer yes/no questions with 99% correctness. They also show that their model generalizes to test data that requires a much longer chain of reasoning with an accuracy of 95+%. Intrigued by their result, our aim in this paper is to further study this approach with respect to reasoning with more general forms of knowledge.

Reasoning about action and change (RAC) is one of the key topic in knowledge representation and reasoning from the early days of AI. The "*going to the airport*" example in (McCarthy 1959) involved reasoning about the effect of the actions "*walk to the car*" and "*drive the car to the airport*". Subsequently McCarthy and others developed the Situation calculus as a tool to reason about actions, with the first paper on this in 1963 (McCarthy 1963) and the seminal paper on it in 1969 (McCarthy and Hayes 1969) where the "*frame problem*" was introduced. Since then there have been many specialized collections (workshop proceedings, journal special issues) on the topics of "*frame problem*" and "*reasoning about actions and change*"; and also several books (Shanahan 1997; Reiter 2001) on the topic. While "*reasoning about actions and change*" continues to be an active area of research in the knowledge representation and reasoning community, recently it has been a highlight aspect in common sense question answering. For example, the ATOMIC knowledge base (Sap et al. 2019) is about actions and their attributes such as (conditional) effects, executability conditions, triggering or preceding conditions, and motivations; and it is used as a key source of commonsense knowledge in the NLQA system (Bosselut et al. 2019).

Thus, in this paper, in our quest to extend the study in (Clark, Tafjord, and Richardson 2020a) to more general forms of knowledge, we explore how well transformers can emulate "*reasoning*" about effects of actions. We explore three example domains, Blocks World, Logistics and DWR (Dock-Worker-Robot) (Ghallab, Nau, and Traverso 2004) which we collectively refer to as the BLD domains, and a generic domain. The Blocks World domain is about a world where blocks are on a table or on top of another block and can be made to towers on the table with the actions being moving a block from its current position to another position. The Logistics domain is about loading/unloading packages from trucks and airplanes and their movements from one position to another. The DWR domain is about a harbor that has locations, cranes, containers, pallets, piles and robots; where pallets can be made to piles, loaded in containers and robots, and moved by cranes; and robots can move from one location to another. The generic domain expresses effects of abstract actions on world properties. We create synthetic worlds of various complexity (based on the number of objects or actions) in these domains and create QA examples with respect to them that we divide into training and test sets. The QA examples includes items with yes/no answers as well as items with somewhat open-ended answers along with numerical values. To create the synthetic examples we use Answer Set Programming (Gelfond and Lifschitz 1988), a declarative language that can express the frame problem in a natural way, and that has many solvers. In our synthetic examples, we focus on the simpler reasoning aspects such as starting with a completely known initial situation whether an action is executable in a particular situation, whether particular properties (referred to as"*fluents*") are true in a particular situation (reached after the execution of a sequence of actions), and count of such properties. Following is an example of QA items created by our synthetic approach:

Initial State: These are locations: fishery, airfield. Robots are: robot-3, r10. Crane are: crane-9, crane-7. There are piles: pile-9, pile-12. These are containers: seashell, moccasin. fishery is adjacent to airfield. fishery has the pile pile-9. pile-12 is at airfield. crane-9 is located at fishery. airfield has crane-7. robot-3 presents at airfield. fishery has r10. robot-3 is unloaded. r10 can hold a container. crane-9 can hold a container. crane-7 can hold a container. These are stacked in order top to bottom : seashell, moccasin. pile-12 has seashell at the top.
Rules: A robot can be at one location at a time. A crane can move a container from the top of a pile to an empty robot or to the top of another pile at the same location. A container can be stacked in some pile on top of the pallet or some other container, loaded on a robot, or held by a crane. A pile is a fixed area attached to a single location. A crane belongs to a single location; it can manipulate containers within that location, between piles and robots. Each robot can carry one container at a time. Robots can move to free adjacent location. A crane is empty if it is not holding container. A robot is unloaded if it is not loaded with container.
Action: Crane-7 picks up seashell and seashell is loaded on r10.
Verify Q:) *Is r10 loaded with moccasin?* Ans: **No**.
Counting Q:) *how many robots are unloaded?* Ans: **1**.
Others Q:) *which container is on top of pile-12?* Ans: **moccasin**.

We pursue various experiments on our synthetic dataset to answer questions and analyze how well a transformer fine tuned with the dataset of (Clark, Tafjord, and Richardson 2020a) can reason about effect of actions in the BLD domains, how well a transformer fine tuned with our training sets does on our test sets, how well it generalizes when the domain is enhanced with more objects, and how well a transformer fine tuned with the generic domain transfers to the specific BLD domains. In our study we consider both explicit specification of rules that express effect of actions as well as the case when these effects are not explicitly given and learned.

Our contributions are summarized below:

1. We provide a framework to study the extent of neural NLQA model's ability to reason about effect of actions. The framework consists of procedurally generated question answering dataset with three types of questions, on four action domain worlds out of which, two are real-world domains. We also provide a hand authored test set.
2. We show RuleTakers (2020a), trained on conjunctive implication rules is unable to generalize to such questions about effect of actions (48-58%).
3. We perform extensive experiments to study the out-of-domain generalization abilities of current state-of-the-art transformer-encoder based QA model.
4. We observe transformer-encoder based QA models can perform reasoning about effect of actions, and do generalize to some extent to out-of-domain worlds (68-90%), but there is still scope of significant improvements.

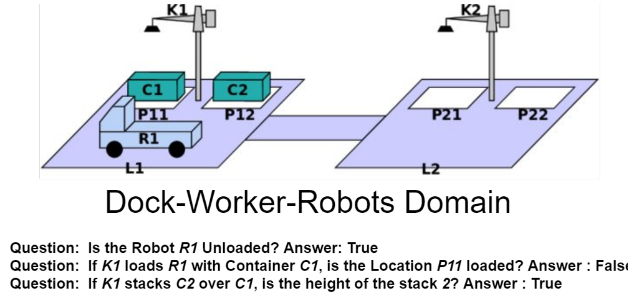

Figure 1: Reasoning about Actions using Question Answering. The world is described in procedurally generated text.

5. We also investigate the ability to learn from a generic domain of actions and fluents, and observe models can somewhat generalize to real world domains (57-83%).

2 BLDG: Domains

In this section, we describe the four action domains, the *blocks world*, *logistics*, *dock-workers robots*, and *generic* domain. The first two domains are well-known benchmarks in the classical automated planning competitions[1]. The third domain is from the text book by Ghallab, Nau, and Traverso (2004). We create the fourth domain to allow using a set-theoretic representation that enables simple generation of instances.

2.1 Blocks World

A blocks world domain consists of a set of named cubes of the same size. A block can be on the table or on top of another block. A block is said to be clear if no other block is on top of it. A clear block can be moved to the table or on top of another clear block. No two blocks can be on the same block at a same time. An *instance* in this domain describes the names and location of the blocks (a.k.a. the *initial state*). Listing 1 describes the effects of actions and their executability conditions of the actions in this domain. In this paper, we use the action language $\mathcal{A}$ proposed by Gelfond and Lifschitz (1993) to represent the action domains.

Listing 1: A Blocks World Domain

```
1  block(a).   block(b).   ...
2  fluent: on(X,Y), ontable(X), clear(X)
3  action: move(X,Y), move(X,table)
4  move(X,Y)  executable_if clear(X), clear(Y)
5  move(X,ta) executable_if clear(X)
6  move(X,Y)  causes on(X,Y),¬clear(Y)
7  move(X,Y)  causes clear(Z) if on(X,Z)
8  move(X,Y)  causes ¬on(X,Z) if on(X,Z)
9  move(X,Y)  causes ¬ontable(X) if ontable(X)
10 move(X,ta) causes ontable(X)
11 move(X,ta) causes clear(Z) if on(X,Z)
12 move(X,ta) causes ¬on(X,Z) if on(X,Z)
13 initially: ontable(b), on(a,b), ...
```

In the above, X, Y, and Z stand for pairwise different blocks and ta stands for table. The first line specificies the objects (blocks) in the instance, the second line defines the

[1]https://www.icaps-conference.org/competitions/

fluents, and the third line defines the actions of the domain. Lines 4 and 5 state the executability of actions. Lines 6–12 encode the effects of the actions, e.g., Line 6 states two unconditional effects of the action of moving the block X on top of the block Y: it will cause X to be on Y and Y no longer clear; Line 7 specifies a conditional effect of the same action: it will clear the block Z if X is currently on Z, etc. The last line specifies the initial state of the world, the block b is on the table and a is on b, etc.

2.2 Logistics

A logistics domain consists of trucks, airplanes, packages at different locations, and different cities. Each location is associated to a city. A location can also be an airport. A package can be loaded to a truck (or an airplane) if the former is at the same location as the latter. Once a package is loaded into a truck (or an airplane), it is inside the truck (the airplane). If a package is inside a truck (an airplane), it can be unloaded from the truck (the airplane). Once unloaded, it is at the same location of the truck (the airplane). A truck can move from one location to another location within a city. An airplane can fly from one airport to another airport. It is assumed that a truck (an airplane) can hold unlimited number of packages. The initial state specifies the location of the trucks, airplanes, and packages.

2.3 Dock-Workers Robots

The dock-workers robots (DWR) domain could be viewed as a combination of the blocks world and the logistics domains without airplanes and airports and trucks are replaced by robots. The domain, taken inspiration from the arrangement of a habor, contains locations, cranes, containers, pallets, piles, and robots. Static information is provided describing the connectivity between locations, the attachment of pallets to locations, and the locations of cranes (that do not move). Robots can move from one location to the adjacenct one if it is not occupied by another robot. Containers are stacked into piles resembling a tower of blocks. A crane can take the container on top of a pile at the same location and holds it. It can also unload the container loaded on a robot at the same location. A crane can load the container it is holding on top of an empty robot or putdown the container on top of a pile at the same location. The initial state specifies the layout of the habor, i.e., the locations of all the objects and other properties (e.g., whether a robot is loaded with a container). Unlike other domains, the DWR domain contains state constraints (a.k.a. static causal laws) which express relationships between fluents[2].

2.4 Generic Domain

In this domain, an action is described by a set of facts of the forms in Listing 2. The first line defines the action $name$. The second line specifies that the executability condition of $name$ is a set called $pre(name)$. The third line declares that the elements of $pre(name)$ are the literals $f1, \ldots, fn$. The fourth line declares that the effect, coded $(name, 1)$ is f. The sixth line specifies the conditions under which the effect $(name, 1)$ is realized, which is the set of literals named $c(name, 1)$ and contains $g1, \ldots, gn$. Similarly, a state constraint is represented by a consequence (e.g., the fluent $occupied(L)$) and a set associated with it (e.g., the set $\{at(R, L)\}$) via a keyword named $static\_law$. We omit the description of static causal laws here for brevity.

Listing 2: Action Representation

```
1 action(name)
2 precondition(name, pre(name), _)
3 member(pre(name),f1) ... member(pre(name),fk).
4 effect(name, e(name,1), f) ....
5 member(c(name,1),g1) ... member(c(name,1),gn)
```

The advantage of this representation is that it enables the generation of random domains with a few parameters such as the number of fluents, the number of actions, the maximal numbers of effects of actions, the number of static causal laws, etc. Observe that this representation can be used to represent the three domains Blocks World, Logistics, and DWR. However, the size of the corresponding representation will increase dramatically, in terms of number of facts. For example, for the Block Worlds domains, with 7 blocks and three towers of 3, 1, 3 blocks, the generic representation has about 1500 facts. Observe that the number of actions and the number of fluents do not change in both representations.

2.5 RAC in Logic Programming

Given an action domain D, represented as a collection of statements about effects of actions and relationships between fluents (Subsections 2.1–2.4), an initial state I, stating the truth value of the fluents in the domain, and an integer n denoting the length of the trajectories that we are interested in, a logic program, $P(D, I, n)$, can be automatically generated (see, e.g., Gelfond and Lifschitz (1998)) that contains rules for

- solving the three fundamental problems in reasoning about effects of actions: the *qualification* problem (when can an action be executed?), the *ramification* problem (how to deal with indirect effects of actions, represented by static laws, e.g., in DWR domain), and the *frame* problem (how to deal with inertial?);
- generating action occurrences:

$$1\ \{occ(A, T) : action(A)\}\ 1 : -time(T), T < n.$$

The answer sets of $P(D, I, n)$ correspond one-to-one to possible evolutions of the world after n actions (a.k.a. the trajectories of length n). Each answer set contains information about the executability of actions at each time step (e.g., whether an action of moving a block a to be on top of the block b can be executed after the execution of moving b on top of a), the truth value of a fluent at a time step k after the action sequence leading to the step k has been executed (e.g., whether the block a is on the table after moving on top of the block b, then moving the block c on top of the block d). This, together with the fact that efficient and scalable ASP solver exists *clingo* (Gebser et al. 2007), allows for the automatic generation of QA examples for RAC by randomly generating the initial state I and using $P(D, I, n)$

[2]The usefulness of state constraints in planning has been discussed in (Thiebaux, Hoffmann, and Nebel 2003; Son et al. 2005)

to generate different worlds. For example, with 20 blocks stacked up to create four towers and the length of 5 actions, generating an answer set (a world) with the corresponding questions can be done in less than 20 second.

3 Synthetic Data Generation

In order to study transformer's capability of reasoning about effects of actions, we generate data for four action domains described in Subsections 2.1–2.4 with three types of questions and different complexities. Each example is a triple (*paragraph*, *question*, *answer*), where *paragraph* provides the initial state, the knowledge of a domain, and a valid action sequence. We generate three types of questions based on the effects of actions to test the model's different reasoning abilities: *verify*, *counting*, and *others*. A **Verify question** is to infer if a statement can be entailed by the paragraph or not (e.g., "Is it possible to move block A on top of block B?"), and answer can be either true or false. A **Counting question** is to count the number of objects satisfying an attribute or a relation (e.g., "How many blocks are on the table?"), and the answer is numeric value. An **Others question** is to reason the status of objects (e.g., "What is the position of block A?"), and the answer is an English word.

Overview We randomly generated the initial states and use the method described in Subsection 2.5 to generate answer sets representing trajectories of length up to 5 for each action domain described in Subsections 2.1–2.4. We use the answer sets to generate questions of the three aforementioned types. To investigate the out-of-distribution (OOD) generalizability of a transformer, for each domain we define five levels of complexity in terms of *novel world* and *action depth*. Novel World complexity is defined by the number of towers for block world domain; the number of airplanes, trucks, and packages for logistic domain; the number of locations, cranes, robots and containers for dock-worker robots domain; the number of fluents, actions, and the complexity of the actions (the maximal number of fluents in the precondition, effects, conditions on each effects) for generic domain. Action Depth complexity is defined by the length of the interested trajectories.

World Generation World description includes the initial state and domain knowledge, where the former is different for each example, and the latter is the same for each example in each domain. The initial state includes two types of facts, attribute $object_i(n)$ (e.g., location(market)) and relation $rel_j(o_1, o_2)$ (e.g., adjacent(market, park)).

The attribute predicates define objects in a domain or describe the status of objects and the relation predicates define the relation between objects. For four domains, the attributes and relation predicates are different:

Blocks world: block/1, ontable/1, clear/1, on/2 ...
Logistic: airplane/1, city/1, truck/1, location/1, inCity/2, ...
DWR: location/1, crane/1, robot/1, pile/1, attached/2 ...
Generic: action/1, fluent/1, effect/3, precondition/3, member/2 ...

where *predicate/n* represents *predicate* is an attribute or relation with n parameters. The domain knowledge includes the effects and constraints of actions mentioned in previous sections.

Question Generation For each domain, we generate three types of questions based on the effects of actions. *Verify* questions are essentially binary questions. We generate "true" questions by entailed effects, (e.g., by $empty(crane_1)$, a true question can be "Is $crane_1$ empty?"). A "false" question can be generated by negating the attribute of objects or replacing an object of a relation with others, (e.g., by $at(robot_1, market)$ in DWR, a false question can be "Is $robot_1$ at airport?"). We balance the number of "true" and "false" questions to avoid label bias. To generate *Counting* questions, we define a set of conditions for four domains, and count how many objects satisfy each condition, e.g. (given a condition in blocks world: executable actions at time 1, and a list of facts $executable(A_1, 1), executable(A_3, 1)$, a counting question with answer 2 can be "How many actions are executable at time 1?"). To generate *Others* questions, we select relation predicates, and mask one object in the question, e.g. (given a relation in logistic: $in(object_1, airplane_1)$, a question with answer $airplane_1$ can be "Where is the $object_1$?"). We have a set of templates with different linguistic variations for each type of questions from which we randomly sample. More examples are present in the supplemental material.

Since we procedurally generate the question-answer, we are able to generate a large number of train and test samples. We divide the train and test samples by first generating a set of unique worlds with a pre-determined initial state configuration, to avoid train-test world overlaps. Further splits are made to test the different complexity worlds.

4 Experiments and Results

4.1 Models

RuleTakers We choose Roberta (Liu et al. 2019) as the pretrained transformer-encoder based question answering model, as it has been demonstrated by Clark, Tafjord, and Richardson to be a near perfect reasoner over natural language rules. We finetune the Roberta model following the recommended hyperparameters in (Clark, Tafjord, and Richardson 2020a) on the associated reasoning dataset. We refer to this model as the RuleTaker. As the provided dataset has only True-False questions, we evaluate it on our Verify questions for all the domains.

Roberta with/without Action-Effect Rules We also train the Roberta model on our generated dataset. We concatenate the initial state and sequence of actions (if any) alongwith the question and train two models, a True-False model similar to the RuleTakers on the binary classification task, and an Open-Ended QA model with a multi-class classification objective, where it learns to generate the answer (Yes/No, Number, Others) similar to the open-ended QA task of visual question answering (Antol et al. 2015).The answer vocabulary for the open-ended QA task is fixed for the train and test splits, and we ensure there are no unknown answers to avoid label shift. Questions are input as: [*CLS*]*context*[*SEP*]*question*[*SEP*], the [CLS] token representation is fed to a feedforward layer to project either to single logit for the binary classification task, or the answer vocabulary size for the open-ended QA task. The action-effect rules

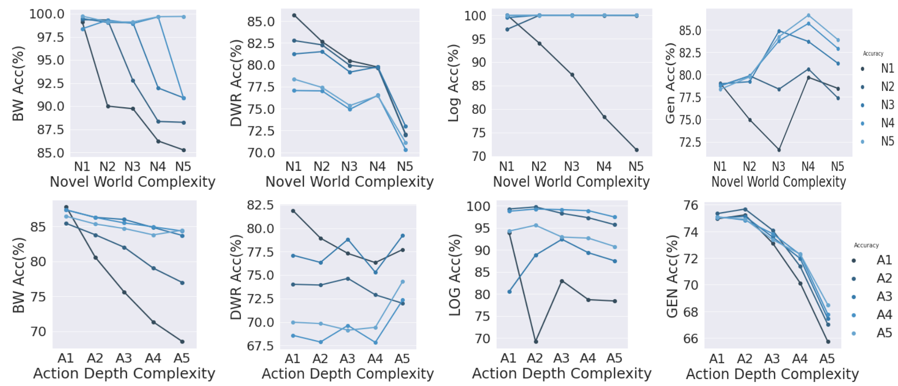

Figure 2: Accuracy trend when trained on simpler worlds and tested on OOD complex worlds **with** action-effect axioms.

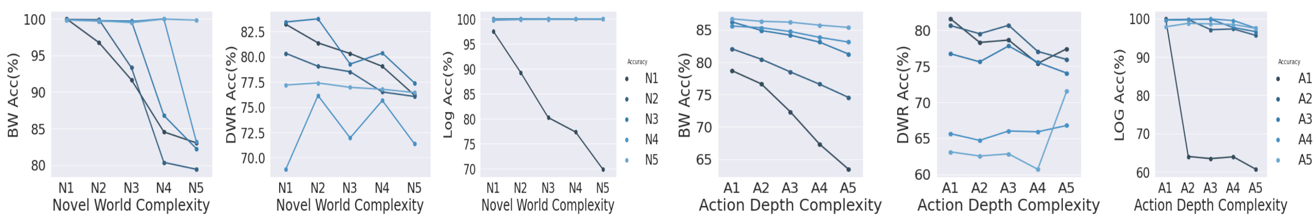

Figure 3: Accuracy trend when trained on simpler worlds and tested on OOD complex worlds **without** action-effect axioms.

Model	Blocks World ↑	Logistics ↑	DWR ↑	Generic ↑
RuleTakers	51.0†	48.0†	58.0†	53.01†
N1 + Rules	<u>85.3</u>	<u>71.3</u>	73.4	78.5
N1 + NoRules	83.0	68.9	<u>76.2</u>	N/A
N3 + Rules	82.3	**99.9**	73.9	**81.3**
N3 + NoRules	**90.9**	**99.9**	**77.4**	N/A
Generic N1	63.4†	57.2†	78.8†	78.4†
Generic N3	64.5†	58.1†	80.6†	81.2†
Generic N5	70.5†	59.6†	83.5†	83.9†

Table 1: Accuracy on T/F questions on OOD novel world (N5) test scenarios. Rules/NoRules indicates with and without action-effect axioms. † evaluated on N1 complexity.

are concatentated with the initial state to create the context. We use the standard cross-entropy loss to train. We evaluate the models using answer exact-match accuracy.

Training & Testing All our experiments are conducted using Roberta-large, which is finetuned on RACE dataset (Lai et al. 2017) using the hyperparameters mentioned in (Liu et al. 2019) as recommended by Clark, Tafjord, and Richardson. We use the same fixed hyperparameters (batch size, learning rate) and use 4 Nvidia V100 16GB gpus.

To test the OOD zero-shot performance we defined the following world complexities, the novel worlds complexities (N1-N5) and action depth complexities (A1-A5). The novel worlds for Blocks world are defined using the number of towers (2,3,4,5,6). In DWR domain the locations, cranes, robots and containers with values for each in (2,3,4,5,6), and similarly for Logistics domain, on airplanes, trucks and packages with values for each in (2,3,4,5,6). In Generic domain the complexity is defined on number of unique actions and fluents with values in range (1-4,5-7,8-10,11-15,16-20). Action depth world complexity is defined the same throughout, with number of actions equal to (1,2,3,4,5). We train all our models with 40K balanced *Verify* questions, 30K *Counting* questions and 30K *Others* questions.

4.2 Results

Can RuleTakers reason about effect of actions? We evaluate the RuleTakers model on our generated *Verify* questions with our action-effect axiom rules. The results are presented in Table 1. We observe that it performs similar to near random in the Blocks World and Logistics domains, and slightly better in the DWR and Generic domains. This demonstrates (unsuprisingly) learning on conjunctive implications is insufficient to reason about effect of actions.

Can Roberta reason about effect of actions? Table 1 summarizes the results on how Roberta finetuned on simpler worlds generalizes to out-of-domain evaluation on larger more complex worlds. We can observe for *Verify* questions, which is the same setting for RuleTakers, it can generalize to novel worlds to some extent. In Table 1, all finetuned models are evaluated on N5, which is the highest complexity, but trained only on N1 and N3. For context, N1 for Blocks World is only 2 towers and N5 is 7 towers, and the world complexity for DWR and Logistics are even more complex. These results indicate that a transformer-encoder based architecture can reason about effect of actions, but there is still scope for further improvements, especially on more complex real-world domains such as DWR.

How does explicit rules affect learning? From Figures 2 and 3, and Table 1 we can observe the effect of action-effect axioms in the three domains. On the novel world complexity OOD test sets, explicit rules show a conclusive positive

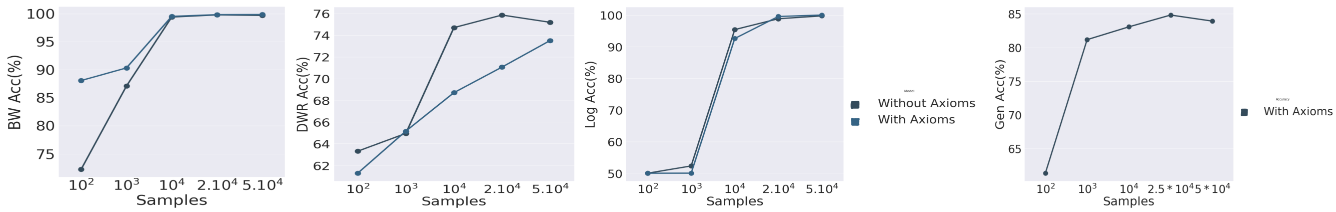

Figure 4: Learning curve trends for different domains on N5 data. For BLD we compare with and without action-effect axioms.

	Blocks World ↑			Logistics ↑			DWR ↑			Generic ↑	
Model	T/F	Num	Other	T/F	Num	Other	T/F	Num	Other	T/F	Num
A1 + Rules	75.13	81.08	47.18	97.98	55.20	75.43	**78.51**	80.30	74.12	87.67	36.47
A1 + NoRules	72.40	76.31	38.61	75.29	33.85	68.12	78.38	78.73	**74.88**	N/A	N/A
A3 + Rules	**92.94**	**94.94**	**60.20**	98.07	65.30	95.60	76.84	**87.51**	74.14	**87.73**	**40.40**
A3 + NoRules	90.20	90.81	59.77	**99.62**	**90.48**	**98.70**	75.95	83.93	61.66	N/A	N/A

Table 2: Accuracy on the three types of questions evaluated on OOD action depth A5 test scenarios.

trend only on the simplest of the domains, i.e, Blocks world. But on the action domain complexity OOD test sets, explicit rules show a more profound effect especially when trained on simple complexity questions (A1-2). For DWR the implicit rules learned generalize better to novel worlds, with a smaller drop in accuracy compared with explicit rules. For Logistics the model trained on smallest complexity performs better with explicit rules. We also observe an interesting phenomenon: If the complexity is more than N2 (3 airplanes, 3 trucks, and 3 packages), the model achieves near perfect results, i.e, it learns to generalize to novel complex worlds (>3 for all). Our hypothesis is that the reasoning needed to answer *Verify* questions may be sufficiently learned on that complexity, unlike Blocks World, or DWR. The Generic domain by definition needs action-effect axioms, and hence cannot be evaluated without it. A model trained on lower complexity domains (lower number of fluents and actions) generalizes well to higher complexity domains (higher number of fluents and actions) but its performance is slightly worse than the performance of its counterpart model trained on higher complexity domains. The model is able to generalize to some extent even with a small number of examples (better than random).

In Figure 4 we observe the positive effect with lower number of samples for Blocks world and Logistics, but a different trend for DWR. Our hypothesis is that some world axioms are simpler to express in natural language, and rules for such domains can show a positive effect in reasoning accuracy. Exploring different ways to express complex domains in natural language and observing its effect on reasoning accuracy will be an interesting future direction.

How many samples are needed to achieve a decent performance? Figure 4 shows that in all the domains the model approaches its best performance on their in-distribution data around 10^4 samples. Even though the rules augmented model's accuracy on the DWR domain appears to improve with increasing training samples till $5 * 10^4$ instances, it does not improve further. It is interesting to observe that other than the Logistics domain, in all other domains the model is able to achieve better than random accuracy (10%+) with a small number of samples.

Can learning on Generic Domain transfer to BLD? We evaluate our models trained on the Generic domain on corresponding mapped questions from BLD domains to verify, if learning about reasoning from actions from a generic domain of actions and fluents transfer to other worlds with complex action spaces. Table 1 summarizes the results. We observe that they do generalize to some extent, statistically significantly above random performance, even when trained on simpler action-fluent complexity. This is interesting to observe and shows the scope of defining and training neuro-symbolic models trained on knowledge representation abstractions to generalize to OOD domains. We evaluate only N1 complexity worlds as the textual description of initial state for N2 and beyond exceeds the 512 token limit of Roberta. The description increases with complexity as the number of unique fluents and actions increase with increase in world complexity. Evaluating on transformers that enable greater than 512 tokens, such as Longformer (Beltagy, Peters, and Cohan 2020), will be an interesting future work.

What happens if we increase the length of action sequence? Table 2 and Figures 2,3 show OOD generaliztion accuracy on different question types, with varying length of action sequences. We observe compared to novel worlds, the action depth complexity OOD generalization is a challenging task. This is expected as the hypothesis space or number of states to track increase with each action taken. One may note that the model does not have an explicit state space where the effect of each individual action is updated, stored and available for reference. Interestingly, models with explicit action-effect rules consistently outperform models that learn implicit rules from examples, especially when trained on less complex worlds. Another observation is that learning on only 1/3 depth action sequence questions, the model is able to generalize to the depth of 5 action sequences on *counting* questions. The counting questions need the model to count the number of fluents that are true in that state. But on the Generic domain, the model's performance is signif-

Model	Blocks World ↑	Logistics ↑	DWR ↑
A3 + Rules	81.0	81.0	75.0
A3 + NoRules	80.4	78.0	77.0
A5 + Rules	**84.6**	**86.0**	**78.0**
A5 + NoRules	83.8	82.0	**78.0**

Table 3: Accuracy on the human authored paraphrased initial states and natural questions in the BLD domains (Overall).

icantly poor, indicating the counting task to be harder. The models comparatively struggle to generalize on the *Others* questions in Blocks world and DWR domains.

How does model generalize to linguistic variations in initial state and questions? We asked expert human annotators to hand author 300 questions of the three types and world initial states for the BLD domains. These questions are used as test sets to evaluate zero-shot transfer on natural paraphrased questions. Its interesting to observe in Table 3 that Roberta is robust to paraphrased and natural questions, including where the objects are different from what it has seen during training (color blocks, to blocks re-named using numbers, animal names).

5 Discussion

A major advantage of our method to create synthetic data using ASP is that we can generate datasets of any size to facilitate learning of complex data-hungry neural models. As we generate answers from ASP solvers, we do not require costly and carefully designed human annotation frameworks to collect answers. Indeed our data generation method suffers with a lack of linguistic diversity, but recent state-of-the-art methods on paraphrasing, and other data augmentation techniques such as back-translation can be used to mitigate that. Moreover similar to the recent development in self-supervised learning techniques, with a shift away from a dataset focussed research, we propose methods to learn from synthetic instances and only evaluating on hand curated real-world representative domains, which avoids the pitfalls of linguistic priors and annotation bias.

There are several advantages of a natural language interface to reason about effect of actions. The requirement of carefully crafted ASP rules to define a domain needs a certain level of expertise in knowledge representation and reasoning. This requirement is avoided when a user can describe a domain in natural language, define the actions and effects, and then query about different states. Learning to perform reasoning about effects of actions from the Generic domain enables generalizing to any novel action domain.

Although we only evaluate Roberta, contrary to previous results shown by Clark, Tafjord, and Richardson, we observe Roberta cannot perfectly answer and generalize to OOD worlds in both novel world complexity and action world complexity. Although ASP solvers with perfect rule definitions can achieve 100% accuracy, but there are two bottlenecks: perfect semantic parsing from text to get the accurate initial state and human involvement in writing perfect rules. Our work is a step towards bridging this gap between these two approaches, by using the expertise of a symbolic "reasoning about action" solver to teach a robust neural model.

6 Related Work and Conclusion

Earlier in Section 1 we mentioned the recent work (Clark, Tafjord, and Richardson 2020a) that motivated our research in this paper and also mentioned some of the prominent works related to reasoning about actions and change. In the context of answering questions with respect to natural language text where rules are explicitly given in natural language the main prior work is (Clark, Tafjord, and Richardson 2020a). That work in its extended version (Clark, Tafjord, and Richardson 2020b) mentions Task 15 of bAbI (Weston et al. 2016), conditional probes in (Richardson et al. 2020), and QuaRTz (Tafjord et al. 2019) among examples of other work where application of general rules given in natural langauge is involved. Some of the tasks in (Weston et al. 2016) and the task in (Mishra et al. 2018) involve reasoning about actions, but in them the effect of actions are *not explicitly stated in natural language*. In (Mitra and Baral 2016) explict answer set programming (ASP) rules are learned from the bAbI dataset and then used to reason using an ASP solver. In this paper we consider both the explicit and implicit cases, i.e., (i) where the effect of actions are explicitly given in natural language, similar to the approach in (Clark, Tafjord, and Richardson 2020a); and (ii) where the effect of actions are not explicitly given but a transformer has to implicitly reason with it. As mentioned in (Clark, Tafjord, and Richardson 2020b) transformers have been shown to be able to (learn to) emulate algorithms in various tasks such as semantic parsing (He and Choi 2019), machine translation (Wang et al. 2019), symbolic integration (Lample and Charton 2019) and mathematics (Saxton et al. 2018). Our work in this paper is perhaps the first work where transformers are tested with respect to a deeper knowledge representation and reasoning challenge, the challenge of reasoning about actions and their effects. In this regard it may be noted that, although reasoning about actions was formally introduced in 1963, it took decades of research before the associated "frame problem" was solved and human researchers were able to come up with formulations, as in (Reiter 1991; Gelfond and Lifschitz 1993), that could reason about actions in a systematic and provable way.

This domain of reasoning about actions and change is the focus of this paper and we show that when trained in a specific domain, transformers are able to reason about the effect of actions in that domain and within that complexity with high accuracy (90-98%). However, when testing with higher complexity the accuracy drops to 68-90%. When trained in the generic domain the transfer learning accuracy with respect to the BLD domains is 57-83%. Thus, even for the simplest aspect of RAC, which is reasoning about effects of actions, further research is needed for better transfer learning / out-of-domain accuracy. Beyond that there are harder challenges in RAC such as planning (Lifschitz 1987), explanation, diagnosis (Baral, McIlraith, and Son 2000) and narrative reasoning (Shanahan 1997; Mueller 2014).

References

Antol, S.; Agrawal, A.; Lu, J.; Mitchell, M.; Batra, D.; Lawrence Zitnick, C.; and Parikh, D. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, 2425–2433.

Baral, C.; McIlraith, S.; and Son, T. C. 2000. Formulating diagnostic problem solving using an action language with narratives and sensing. In *KR*, 311–322.

Beltagy, I.; Peters, M. E.; and Cohan, A. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150* .

Bosselut, A.; Rashkin, H.; Sap, M.; Malaviya, C.; Celikyilmaz, A.; and Choi, Y. 2019. Comet: Commonsense transformers for automatic knowledge graph construction. *arXiv preprint arXiv:1906.05317* .

Clark, P.; Tafjord, O.; and Richardson, K. 2020a. Transformers as soft reasoners over language. In *IJCAI*.

Clark, P.; Tafjord, O.; and Richardson, K. 2020b. Transformers as soft reasoners over language. *arXiv preprint arXiv:2002.05867* .

Gebser, M.; Kaufmann, B.; Neumann, A.; and Schaub, T. 2007. clasp: A Conflict-Driven Answer Set Solver. In Baral, C.; Brewka, G.; and Schlipf, J., eds., *Proceedings of the Ninth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'07)*, volume 4483 of *Lecture Notes in Artificial Intelligence*, 260–265. Springer-Verlag.

Gelfond, M.; and Lifschitz, V. 1988. The stable model semantics for logic programming. In *ICLP/SLP*, volume 88, 1070–1080.

Gelfond, M.; and Lifschitz, V. 1993. Representing action and change by logic programs. *The Journal of Logic Programming* 17(2-4): 301–321.

Gelfond, M.; and Lifschitz, V. 1998. Action Languages. *Electronic Transactions on Artificial Intelligence* 3(6).

Ghallab, M.; Nau, D.; and Traverso, P. 2004. *Automated planning: theory and practice*. Morgan Kaufmann Publishers.

He, H.; and Choi, J. D. 2019. Establishing strong baselines for the new decade: Sequence tagging, syntactic and semantic parsing with bert. *arXiv preprint arXiv:1908.04943* .

Lai, G.; Xie, Q.; Liu, H.; Yang, Y.; and Hovy, E. 2017. RACE: Large-scale ReAding Comprehension Dataset From Examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 785–794.

Lample, G.; and Charton, F. 2019. Deep Learning For Symbolic Mathematics. In *International Conference on Learning Representations*.

Lifschitz, V. 1987. On the semantics of STRIPS. In *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop*, 1–9.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* .

McCarthy, J. 1959. *Programs with common sense*. RLE and MIT computation center.

McCarthy, J. 1963. Situations, actions, and causal laws. Technical report, Stanford University.

McCarthy, J.; and Hayes, P. J. 1969. Some philosophical problems from the standpoint of artificial intelligence. Technical report, Stanford University.

Mishra, B. D.; Huang, L.; Tandon, N.; Yih, W.-t.; and Clark, P. 2018. Tracking state changes in procedural text: a challenge dataset and models for process paragraph comprehension. In *NAACL*.

Mitra, A.; and Baral, C. 2016. Addressing a Question Answering Challenge by Combining Statistical Methods with Inductive Rule Learning and Reasoning. In *AAAI*, 2779–2785.

Mueller, E. T. 2014. *Commonsense reasoning: an event calculus based approach*. Morgan Kaufmann.

Reiter, R. 1991. The Frame Problem in the Situation Calculus: A Simple Solution (Sometimes) and a Completeness Result for Goal Regression. In *Artificial and Mathematical Theory of Computation*, 359–380. Citeseer.

Reiter, R. 2001. *Knowledge in action: logical foundations for specifying and implementing dynamical systems*. MIT press.

Richardson, K.; Hu, H.; Moss, L. S.; and Sabharwal, A. 2020. Probing Natural Language Inference Models through Semantic Fragments. In *AAAI*, 8713–8721.

Sap, M.; Le Bras, R.; Allaway, E.; Bhagavatula, C.; Lourie, N.; Rashkin, H.; Roof, B.; Smith, N. A.; and Choi, Y. 2019. Atomic: An atlas of machine commonsense for if-then reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 3027–3035.

Saxton, D.; Grefenstette, E.; Hill, F.; and Kohli, P. 2018. Analysing Mathematical Reasoning Abilities of Neural Models. In *International Conference on Learning Representations*.

Shanahan, M. 1997. *Solving the frame problem: a mathematical investigation of the common sense law of inertia*. MIT press.

Son, T. C.; Tu, P. H.; Gelfond, M.; and Morales, R. 2005. Conformant Planning for Domains with Constraints — A New Approach. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, 1211–1216.

Tafjord, O.; Gardner, M.; Lin, K.; and Clark, P. 2019. Quartz: An open-domain dataset of qualitative relationship questions. In *EMNLP/IJCNLP*.

Thiebaux, S.; Hoffmann, J.; and Nebel, B. 2003. In Defense of PDDL Axioms. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03)*.

Wang, Q.; Li, B.; Xiao, T.; Zhu, J.; Li, C.; Wong, D. F.; and Chao, L. S. 2019. Learning deep transformer models for machine translation. *arXiv preprint arXiv:1906.01787* .

Weston, J.; Bordes, A.; Chopra, S.; Rush, A. M.; van Merriënboer, B.; Joulin, A.; and Mikolov, T. 2016. Towards ai-complete question answering: A set of prerequisite toy tasks. In *ICLR*.