Exploring the Capacity of Pretrained Language Models for Reasoning about Actions and Change

Weinan He[1], **Canming Huang**[1], **Zhanhao Xiao**[2]*, **Yongmei Liu**[1]*
[1]Dept. of Computer Science, Sun Yat-sen University, Guangzhou 510006, China
[2]School of Computer Science, Guangdong Polytechnic Normal University,
Guangzhou 510665, China
{heweinan, huangcm}@mail2.sysu.edu.cn, xiaozhanhao@gpnu.edu.cn,
ymliu@mail.sysu.edu.cn

Abstract

Reasoning about actions and change (RAC) is essential to understand and interact with the ever-changing environment. Previous AI research has shown the importance of fundamental and indispensable knowledge of actions, i.e., preconditions and effects. However, traditional methods rely on logical formalization which hinders practical applications. With recent transformer-based language models (LMs), reasoning over text is desirable and seemingly feasible, leading to the question of whether LMs can effectively and efficiently learn to solve RAC problems. We propose four essential RAC tasks as a comprehensive textual benchmark and generate problems in a way that minimizes the influence of other linguistic requirements (e.g., grounding) to focus on RAC. The resulting benchmark, **TRAC**, encompassing problems of various complexities, facilitates a more granular evaluation of LMs, precisely targeting the structural generalization ability much needed for RAC. Experiments with three high-performing transformers indicate that additional efforts are needed to tackle challenges raised by TRAC.

1 Introduction

Reasoning about actions and change (RAC) has been a central issue since the early days in AI (McCarthy, 1963) and received much attention from the NLP community (Li et al., 2021; Zellers et al., 2021; Dalvi et al., 2018). Classical AI has long recognized the importance of **actions** and characterized the fundamental knowledge about actions: **preconditions** and **effects** (Reiter, 2001). Preconditions are the conditions that must be satisfied when actions are executed, while effects define the result. Consider the example where an agent is moving containers in a dock: Suppose a red container is on top of a green one, and the agent is given an instruction to "move the green container to another platform". To achieve such a goal, it needs to know the preconditions of moving a container (there is not any other container on top of it) and the effect of such actions.

While traditional approaches provide sound and effective reasoning by capturing preconditions and effects, they rely on expensive and difficult formalization. Consequently, inducing knowledge from text and reasoning directly is becoming more preferable. As transformers have shown their promising potential in many linguistic tasks (Liu et al., 2019; Raffel et al., 2020), we set out to probe whether pretrained language models can master the ability of reasoning about actions and changes.

Previous approaches involving actions in NLP are more *application*-centric and usually contain specific tasks (e.g. instruction following, prediction) (Li et al., 2021; Zellers et al., 2021) without considering both preconditions and effects. In general these tasks are not verified logically. On the other hand, there exist some template-based datasets, such as the one in (Clark et al., 2020), which are logically sound and generated from several simple rules. Unfortunately, they pay more attention to general deduction in a static perspective, while RAC problems require repeated changes.

We propose **Textual Reasoning about Actions and Change** (**TRAC**), a comprehensive suite of four fundamental and granular RAC reasoning tasks, inspired by traditional reasoning problems (Brachman and Levesque, 2004). Recognizing the utmost importance of preconditions and effects, we include the two fundamental tasks, **Projection** and **Executability**, which *directly target* the *essential knowledge* of RAC. We also provide two composite tasks, **Plan-Verification** and **Goal-Recognition** for more *comprehensive* problem settings. Together, the four aspects of TRAC enable a more granular evaluation. We position TRAC as a diagnostic benchmark that is a more direct and precise embodiment of RAC abilities:

[*] Corresponding author

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 4629–4643
July 9-14, 2023 ©2023 Association for Computational Linguistics

- **Projection**: anticipate the effects of actions;
- **Executability**: decide if actions are applicable;
- **Plan-Verification**: decide if actions form a valid plan;
- **Goal-Recognition**: recognize the goal from observations of actions.

We employ two principles in dataset construction. Firstly, we minimize the impact of other important abilities (e.g. grounding and language variance) for a “clean-room” evaluation that solely focuses on RAC. Secondly, we desire reasoning problems with controlled complexities for testing structural generalization. We design a framework that takes the action domain knowledge and the textual template as input, generates symbolic problems, and synthesizes the textual problems. In this paper, we select the blocks world (Cook and Liu, 2002), a typical action domain for a proof-of-concept. However, the framework is domain-agnostic and is thus extensible to other action domains.

We design generalization tests based on the observation that once the classical AI systems are endowed with knowledge of actions, they can effectively solve structurally more complex problems. Therefore, we lay out four aspects of generalization experiments: 1) with more objects; 2) with longer action sequences; 3) with unseen names of objects; 4) with unseen conjunctive conditions. We train neural language models (LMs) of three high-performing transformer architectures, testing their efficiency and effectiveness on the TRAC datasets. The result shows that while transformers are able to induce and utilize knowledge from a large number of training examples, it remains a great challenge to efficiently generalize to structurally more complex problems on TRAC.

2 Related Work

Recent NLP tasks are embracing reasoning. While machine reading tasks such as SQuAD (Rajpurkar et al., 2018) require inference to some extent, our work is the first one as we know to systematically study Textual RAC. Reasoning about dynamic worlds is also needed in commonsense tasks such as the Winograd Schema Challenge (Levesque et al., 2012; Wang et al., 2019). But environments in such tasks did not model well-defined actions and change. Compared to these tasks, our concern is formal reasoning that supports sound conclusions.

Our work is distinct from previous attempts of formal reasoning in natural language. Natural Logic studies valid inference over natural language (Lakoff, 1970), but its recent developments focus on reasoning about semantic relations between lexical terms (Angeli et al., 2016). Meanwhile, transformers have achieved promising results on textual deduction tasks (Clark et al., 2020; Saha et al., 2020). While they target general deduction such as “round people are nice; Bob is round; Thus Bob is nice”, we dedicate our work to RAC and requires systems to *learn* the knowledge from examples.

Previous work has been exploring NLP tasks involving actions, especially with instruction following, action outcome prediction and procedural text generation tasks. TRAC is unique in its focus on fundamentals, broad coverage of reasoning tasks, and the granularity. Linguistic requirements (e.g. the ability of grounding and handling languge variance), though important, are foreign to RAC and are thus avoided. Compared to application-centric tasks, our work provides a more abstract view of tasks that embody the fundamental requirements of RAC. Moreover, this paper proposes a suite of generalization tests that target structural complexities in RAC, which enables more fine-grained tests.

Instruction Following (IF): Agents are given textual directives to execute actions in an environment. In (Zhou et al., 2021; Dan et al., 2021), agents receive visual features or spatial coordinates as input. More importantly, IF does not target directly preconditions and effects as we do.

Prediction: Projection and prediction both concern the effect of actions. In (Zellers et al., 2021), symbolic attributes represent the world instead of language. Questions in bAbI (Weston et al., 2016) and ProPara (Dalvi et al., 2018) ask about change of objects. However, they neglect the equally important preconditions.

Procedure Generation: Predicting the next instruction (Li et al., 2021) or next step (Bosselut et al., 2018) might relate to the executability of actions, but they only cover partial RAC illustrated in this paper. While understanding the preconditions is needed, it could also implicate other extraneous factors such as the preferences of the instructor.

Generalization has always been a desirable trait. In (Dan et al., 2021), the authors attacked IF systems with adversarial examples (e.g. with slight perturbations). Long-horizon problems are shown to be non-trivial for neural planners (Zhou et al., 2021; Xu et al., 2019). Our evaluation with transformers further corroborate the observation. Additionally, TRAC imposes more structural generalization requirements in a more granular way.

3 TRAC

In this section, we first introduce the theoretical background before we lay out the four tasks and discuss dataset generation. The dataset is available at `https://github.com/sysulic/trac`.

3.1 Theoretical Preparation

In RAC, the essential issues include understanding 1) *the state of the environment*, 2) *which actions are applicable*, and 3) *how actions affect the environment*. For precise definitions of states and actions, we use the semantics of STRIPS (Fikes and Nilsson, 1971), a typical formal language that enables sound reasoning. In STRIPS, an *action domain* specifies the types of objects, predicates for describing the states, and how actions change the environment. In our exploration, we limit the action domains to be *deterministic* and *noise-free*. We also assume the *unique name axioms*, where different objects have different names. TRAC is built upon the following concepts:

- A **state** is a set of atomic expressions (e.g., $\{clear(A), on(A, B), \ldots\}$) that represents a snapshot of the environment at a specific time point. Atoms not in the state are considered false.
- An **action** consists of four parts: name, precondition, add list and delete list, the last three of which are sets of atoms. An action is applicable to a state iff the atoms of the precondition are all contained in the state. When it is applied to a state s, expressions in the delete list will be removed from s and those in the add list will be added into s, resulting in a new state s'.
- An **action sequence** is an ordered sequence of actions. It is applicable to a state if and only if every action can be applied consecutively.
- A **goal** is a ground formula that describes the objective state. We limit goals to be either a literal (an atom or its negation) or a conjunction of two. We only consider achievable goals.
- A **plan** (wrt a goal and a state) is an applicable action sequence. It is *goal-achieving* if the execution of the plan achieves the goal. A goal-achieving plan is *optimal* if there is no shorter plan.

We use a variant of the blocks world (BW) as an example where 1) a table has infinite space to hold blocks, 2) blocks have identical sizes and can be stacked as towers (a block is either on another one or immediately on the table), and 3) a block can be moved only if certain conditions are met. Example 1 and 2 show the predicates and actions in BW. Given an action domain and an initial state, the action and expression space can be determined, based on which the TRAC problems are generated.

Example 1. Predicates in BW:

1. $on(x, y)$ states that x is on y;
2. $onTable(x)$ says x is on the table;
3. $clear(x)$ declares that there is no block on x.

Actions in BW. We provide the definitions of the actions.

Example 2. Actions in BW:

$move(x, y, z)$: Move block x that is on block y onto block z.

- Precondition: $clear(x), clear(z), on(x, y)$.
- Add list: $clear(y), on(x, z)$.
- Delete list: $clear(z), on(x, y)$.

$moveToTable(x, y)$: Move block x that is on block y onto the table.

- Precondition: $clear(x), on(x, y)$.
- Add list: $onTable(x), clear(y)$.
- Delete list: $on(x, y)$.

$moveFromTable(x, y)$: Move block x that is on the table onto block y.

- Precondition: $onTable(x), clear(x), clear(y)$.
- Add list: $on(x, y)$.
- Delete list: $onTable(x), clear(y)$.

3.2 Reasoning Tasks in TRAC

For a comprehensive and granular evaluation, we propose four different reasoning tasks, each focusing on an aspect of RAC. All four tasks in TRAC are formulated as text classification problems, similar to the deduction task (Clark et al., 2020) and Natural Language Inference task. Given the input of two texts, a *context* and a *query*, the system is asked to classify if the query is true accordingly. Concrete examples can be seen in Table 1.

Projection. The projection task directly asks about the effects of actions: Given an initial state s and an applicable sequence $\vec{a}$ of N actions, decide whether the projection query q, a proposition, would hold after the execution of $\vec{a}$. The context is s and $\vec{a}$, and the query is q.

Task	Context	Query	Answer
PR	s: The green block is on the table. The red block is clear. The blue block is clear. The green block is clear. The red block is on the table. The blue block is on the table. $\vec{a}$: Jane moves the green block from the table to the red block.	q: The blue block is on top of the red block.	False
EX	s: The olive block is on the table. The yellow block is on top of the olive block. The indigo block is clear. The indigo block is on top of the yellow block.	$\vec{a}$: Jane moves the indigo block from the yellow block onto the table.	True
PV	s: The blue block is clear. The blue block is on top of the magenta block. The magenta block is on top of the white block. The white block is on the table. g: the blue block is not on top of the magenta block	$\vec{a}$: Jane moves the blue block from the magenta block onto the table.	True
GR	s: The blue block is clear. The blue block is on top of the magenta block. The magenta block is on top of the white block. The white block is on the table. $\vec{a}$: Jane moves the blue block from the magenta block onto the table.	g: the blue block is on top of the magenta block.	False

Table 1: Examples of TRAC tasks. Each problem in TRAC consists of a context, a query, and an answer. The markers $s, \vec{a}, g$ are only shown here for reference. (PR=Projection, EX=Executability, PV=Plan-Verification, GR=Goal-Recognition)

Executability. This task directly targets the preconditions of actions: Given an initial state s and a sequence $\vec{a}$ of N actions, decide whether $\vec{a}$ can be executed consecutively in s. The context is s and the query is $\vec{a}$.

Plan-Verification (PV). Planning is the task of formulating actions to fulfill a certain goal. In TRAC, we use the verification version that asks systems to recognize if the provided actions can achieve the goal: Given an initial state s and a goal g, a proposition, and a sequence $\vec{a}$ of N actions, decide if a can achieve g. The context is s and g, and the query is $\vec{a}$.

Goal-Recognition (GR). GR is the task to recognize the goal from the partial observation of actions. We use a simplified version, where systems observe a partial action sequence and need to figure out if the given goal is the true objective: Given an initial state s, a potential goal g, and a sequence $\vec{a}$ of N actions as the observation, decide if g is the true objective. That is, decide if $\vec{a}$ is a prefix of any optimal plans to achieve g. The context is s and $\vec{a}$, and the query is g.

3.3 Dataset Generation

We provide a framework to generate TRAC problems. It takes both the action domain and the language template as input to construct symbolic forms and translation, respectively. For each task in TRAC, we generate three basic datasets, each having action sequences of different lengths (denoted as L1, L2, and L3 for lengths 1, 2, and 3 respectively). All examples in the datasets have $M = 5$ objects. For the generalization tests discussed in detail in the next section, we also construct additional datasets with different parameters. Each problem is first generated in the symbolic form before being translated into textual form in English.

Symbolic Instance Generation. Commonly existing in all examples are the *initial state* and the *action sequence*, which serve as the foundation for all four tasks. While the initial state is always part of the context, the action sequence is either part of the context or is the query, depending on the task. Firstly we generate the *state space* with M objects according to the action domain. In the blocks world where blocks have different colors, their names (e.g. "the red block") are randomly chosen from a pre-specified range. Secondly, the action space is computed, which includes all grounded possible actions with respect to each possible state. With these spaces, we construct the context and query for each TRAC problem.

Projection. The context includes an initial state and an action sequence. From the action space, we randomly sample N actions to form a sequence that is executable in the initial state. For the query, we randomly generate a formula of the following form:

$$l_1 \text{ or } l_1 \wedge l_2, \tag{1}$$

where l_1 and l_2 are literals (atoms or their negations), e.g., $onTable(Blue)$, $\neg on(Green, Blue)$. The query is true if and only if it holds after executing the action sequence.

Executability. The context is an initial state. For the query, we sample N actions from the action space as a sequence. The query is true if and only if it is executable *consecutively* in the initial state.

Plan-Verification. The context consists of an initial state and a goal that is achievable with N actions or less. The query is an action sequence of length N. Both the goal and the action sequence are generated at random. Goals share the same form as projection queries, shown in Formula 1. We only include achievable goals in this task. The query is true if and only if it is a valid goal-achieving plan.

Goal-Recognition. The context comprises an initial state and a sequence of N actions as the partial observation. The GR query is a plausible goal of the same form as in the plan-verification task. Both the action sequence and the goal are randomly generated. The query is true if and only if the observation is a prefix of any optimal plans that achieve the query.

Textual Form Synthesis. We use templates to generate the actual problems, following the guideline of "clean-room" evaluation, which strives to focus on RAC instead of other linguistic requirements such as grounding and language variance. To maintain readability, our framework utilizes handcrafted templates for the specific action domain. These templates specify the translation of each predicate, action, goal, and projection query. A symbolic state will be converted into several sentences, each of which is a direct translation of the atomic expression. Similarly, an action sequence is translated into a concatenation of the action sentences. Projection queries are synthesized in the same fashion, but goals are processed differently to accommodate conjunctions with "and" if necessary. For example, the textual form of the goal $onTable(Blue) \wedge \neg on(Green, Blue)$ is "the blue block is on the table and the green block is not on the blue block".

As a result, we generate four datasets (four tasks, each containing various lengths of action sequences). Each dataset contains 15,000 label-balanced examples. We split the 15k examples into 12k training examples (where 2k are used as a dev set) and 3k testing examples.

4 Experiments

Task	RoBERTa	GPT-2	T5
PR	**87.36** (0.0396)	85.13 (0.0336)	82.99 (0.0227)
EX	**99.73** (0.0013)	99.37 (0.0037)	98.83 (0.0024)
PV	87.63 (0.0158)	**90.09** (0.0157)	87.73 (0.0110)
GR	96.82 (0.0044)	**97.44** (0.0021)	94.04 (0.0082)

Table 2: Accuracies (percent signs omitted) and standard deviations (in parentheses) of the baselines on TRAC. Each cell corresponds to the model trained and tested on the specific dataset (column header) for the task. (PR=Projection, EX=Executability, PV=Plan-Verification, GR=Goal-Recognition)

We conduct experiments to address the following questions:

1. Can transformers induce knowledge to effectively solve TRAC problems?
2. Can they generalize to problems that are structurally more complex?
3. How data-efficient are they?

The datasets, code, and hyper-parameters are available in the supplementary materials.

4.1 Baseline Models

We use three different LMs as our baseline models, each with different architectures: RoBERTa (Liu et al., 2019), GPT-2 (Radford et al., 2019), and T5 (Raffel et al., 2020). These architectures compose transformer layers in various typical fashions:

1. RoBERTa features transformer-encoder layers;
2. GPT-2 contains transformer-decoder layers;
3. T5 combines both encoder and decoder layers.

Driving by the classification nature of TRAC, the first two baseline models are built by adding a linear layer upon the transformer layers for both

Task	SD	GE1	GE2		GE3	GE4	
			L4	L5		Literals	Conj.
PR	87.36 (0.0396)	58.19 (0.0185)	71.91 (0.0428)	69.82 (0.0301)	85.09 (0.0308)	93.15 (0.0537)	72.89 (0.0248)
EX	99.73 (0.0013)	87.91 (0.0469)	82.54 (0.0200)	79.76 (0.0159)	99.01 (0.0014)	N/A	N/A
PV	87.63 (0.0158)	80.40 (0.0461)	61.67 (0.0169)	56.27 (0.0086)	91.81 (0.0181)	98.11 (0.0014)	68.63 (0.0441)
GR	96.82 (0.0044)	79.66 (0.0475)	N/A	N/A	94.69 (0.0040)	99.99 (0.0001)	73.91 (0.0028)

Table 3: Accuracies and standard deviations of the RoBERTa-base models on generalization experiments. Results from Table 2 are shown in the second column (SD=Standard) as a comparison for GE1, GE2, and GE3. The last two columns report results for GE4: the baselines are trained using examples with only literals. (PR=Projection, EX=Executability, PV=Plan-Verification, GR=Goal-Recognition)

RoBERTa and GPT-2. As for T5, we use its text-to-text pre-training objective to generate the labels. The input for RoBERTa is organized as <s> `context </s> query </s>` where <s> and </s> are separator symbols, following previous work (Clark et al., 2020). We tarin these LMs to predict the truth of the query using the cross-entropy loss function. Accuracy is used as the metric for evaluation, and we report the mean values and standard deviations of the repeated experiments. Appendix A.4 gives details of the LMs.

4.2 Effectiveness of Transformers

We first evaluate the effectiveness of the baseline models on TRAC problems of the same structural complexity. Both the training set and the test set contain problems with $M = 5$ objects and N actions ($N \in \{1, 2, 3\}$, where the ratio of L1:L2:L3 is 1:1:1). For each architecture, we train a baseline model separately for each reasoning task, resulting in twelve such models, shown in Table 2. In this setting, transformers are given enough training data and are required to induce knowledge about actions and change from examples.

In Table 2, transformers have shown capable performances, with all accuracies above 80%. While they excel at Executability and Goal-Recognition, there is considerable room for improvement on Projection and Plan-Verification. Although these different transformer architectures have their own wins on different tasks, they do have rather similar performances, demonstrating that the challenge of TRAC is universal for transformers.

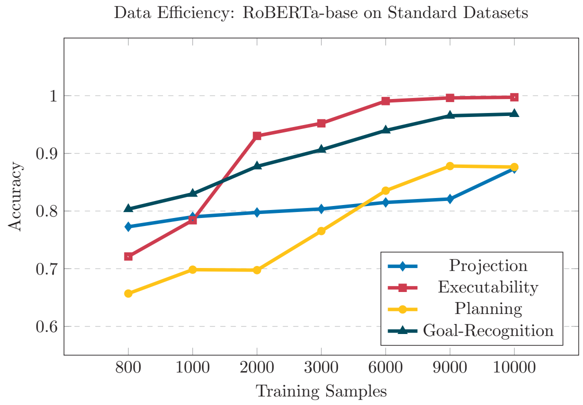

Figure 1: Accuracies of RoBERTa baselines vs sizes of training samples on the standard datasets of TRAC.

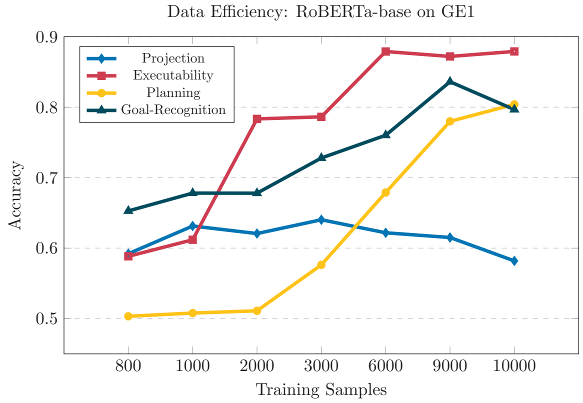

Figure 2: Accuracies of RoBERTa baselines vs sizes of training samples on the GE1 of TRAC.

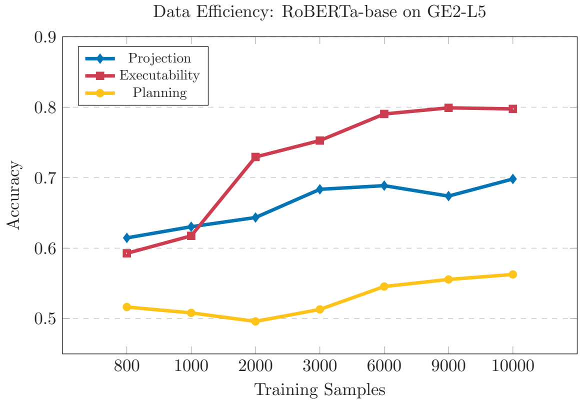

Figure 3: Accuracies of RoBERTa baselines vs sizes of training samples on the GE2 of TRAC.

4.3 Structural Generalization Experiments

Targeting structural generalization ability, we design four out-of-distribution tests. Novel test examples that are more structurally complex are generated to this end. For the first three generalization experiments (GEs), we re-use models trained on previous datasets (with complexities $M = 5, N \in \{1, 2, 3\}$). Since the last GE requires additional training, both training and test sets are necessary. In total, we created twenty additional datasets beyond the normal ones. Table 3 shows the results for the RoBERTa-base models. The other two tranformer architecture are also evaluated.

GE1: More Objects. In this test, we examine whether the baselines can handle TRAC problems containing more objects (blocks in the BW domain). We generate a new dataset for each reasoning task, all of which involve ten objects. We evaluate the baselines trained on the standard datasets. The results show that baselines have significantly worse performances on GE1 datasets for all tasks, most notably in Projection. This is expected, as the GE1 problems have longer state descriptions and more complicated states. Compared to the training examples, the state descriptions in GE1 datasets contain 6.1 more sentences with 52.1 more words on average. Nonetheless, the results show that the baselines do not generalize well to more objects.

GE2: More Actions. Naturally, we are interested in the generalization to longer action sequences, as the length often plays a vital role in both formal reasoning tasks and natural language tasks (Zhou et al., 2021; Xu et al., 2019). For Projection, Executability, and Plan-Verification tasks, we generated the L4 and the L5 datasets that have action sequences of length four and five, respectively. Goal-Recognition is left out as there are not enough test examples in this setting. The results from Table 3 confirm that the length of the action sequence is a vital factor, as the accuracies degrade for longer sequences. It is more apparent for Plan-Verification problems, with an appalling 40% accuracy loss on L5. This observation further verifies the universality of the long-horizon problem (Xu et al., 2019), even for transformer-based LMs.

GE3: Unseen Names. Changing the names of the objects is supposed to have negligible impact. The ability of generalization to unseen symbols is desirable in reasoning over both formal and natural languages. Seeing this, we substitute the names of objects for previously unseen ones, resulting in four more datasets. They differ from the standard ones only in the object names. The results from Table 3 show minor differences between the performances as expected, demonstrating the capability of the LMs to generalize beyond unseen names.

GE4: Compositionality in Goals and Projection Queries. Reasoning requires the capability of *compositionality*: to understand or manipulate higher-level structures composed of known components. One such ability is to combine two conditions as a conjunction. If systems understand conditions A and B separately, they are expected to realize that the conjunction $A \wedge B$ is true iff *both* are true. In TRAC tasks, the ideal testbeds are Projection, Plan-Verification, and Goal-recognition, where the projection queries and goals could be partitioned into literals and conjunctions.

In Projection, conjunctions take the form of two separate sentences. For example, a projection query could be “The red block is on top of the green block. The green block is on the table.” Whereas in Plan-Verification and Goal-Recognition, conjunctions have the specific “and” surface form. For example, a goal could be “the red block is clear and the red block is on top of the green block”.

In this experiment, we train baselines using examples with only literals in conditions and see if they can handle the examples with conjunctions. Therefore, new training datasets are needed. For each of the three tasks, we generate GE4-literals, a dataset of 15k instances with only literals in target conditions (10k for training, 2k as dev set, and the other 3k for testing), along with GE4-conjunctions, a dataset of 3k problems with only conjunctions.

After training the baselines on GE4-literals, we compare their performances on test sets of literals and conjunctions.

The results from Table 3 show that the baselines do not generalize well to conjunctions, losing more than 20% accuracies on all tasks. Such a phenomenon suggests that compositionality is not trivial in TRAC for the transformers. It is also noteworthy that conjunctive conditions have various forms: While conjunctions are represented by two sentences in Projection, they are of the form “condition1 and condition2” in Plan-Verification and Goal-Recognition. This leads us to believe that the performance loss is not about the introduction of the surface form “and”, but about the conceptual understanding of conjunctive compositionality, without which the models cannot generalize well.

4.4 Data Efficiency

In reality, humans need only a few examples to adapt to novel environments. To explore how many training samples are needed for the transformers, we plot the accuracies of the RoBERTa baselines with increasing numbers of training samples. In Figure 1, we notice that the baselines require at least 3000 samples to have acceptable accuracies (above 80%) on standard datasets. The inefficiency is even more obvious when it comes to GE1 and GE2 examples in Figure 2 and Figure 3, respectively. Moreover, the Plan-Verification task seems to be the most challenging one when training data is limited.

Other Transformers. We also evaluate GPT-2, T5, and a larger RoBERTa model on the GEs and the full results are provided in Appendix A.1.

- GPT-2 and T5 have rather similar performances on GEs, suggesting that structural generalization is a universal challenge for transformers;
- Although the larger RoBERTa model outperforms smaller models, it also suffers when facing structurally more complex problems.

5 Discussion

Although we are optimistic about the future of transformers, their performances in the generalization tests indicate that transformers alone are not enough for RAC: Firstly, the reasoning problems in this proof-of-concept evaluation involve few actions objects that would be quite effortless for humans. Secondly, the scale of generalization is rather minor (from three actions to four or five actions; from five objects to ten objects). Yet we could observe the struggle of transformers with such minute structurally complex problems. As illustrated in (Li et al., 2021), transformers capture meaning in texts to some extent, which indicates that they have potential in modeling actions and change. Such potentials can also be seen in our experiments when transformers are given more than abundant training examples. Meanwhile, (Zellers et al., 2021) showed that dedicated neural components other than transformers could help model state changes and predictions. These lead us to the conjecture that additional mechanisms that model the **preconditions** and **effects** could be the next objectives towards solving TRAC problems.

6 Limitations

More Complex Domains. We choose the BW for its intuitive and simplistic nature (with one kind of object, three types of actions, and three predicates). Although the generalization experiments suffice currently to challenge transformers, real-world situations are more complicated. With the improvement of the algorithms, the need for a better arrangement of actions domains is emerging. In time, it could be beneficial to include several domains with various levels of complexities.

Balance Between Rigor and Natural. For now, the synthesized English sentences are generated using a fixed template. Whilst being accurate without ambiguity, the resulting text is still quite formal. It would be valuable to add variety in the expressions without losing precision.

Better Solvers. As our demonstrations suggest, current LMs still fall short on the generalization tests. We hope that our work will pique interests in the community towards reasoning about actions and change, and challenge approaches to undertake the fundamental reasoning tasks.

7 Conclusion

In a time where language models excel at many natural language tasks, including deductive ones, we revisit the key reasoning abilities for dynamic worlds with actions and change. While preserving the essence of traditional formal reasoning, we set out to investigate how well transformers can reason rigorously over textual input, which avoids the need for a complete formalization of each specific problem. Using the semantics of STRIPS, we characterize four essential reasoning tasks about actions and change to form the TRAC benchmark. We devise a framework to generate symbolic problems and transform them into text, resulting in a suite of datasets of various complexities. We also design four further experiments that target different aspects of structural generalization. Built upon the high-performing transformers, the baselines are put to the test under different settings. Although they show promising results on in-distribution problems provided with more-than-abundant training examples, it is the out-of-distribution generalization tests that cause troubles. We argue that TRAC tasks could be used to 1) expand our understanding of the limitations of transformers and, 2) serve as a challenge for generalization in RAC over text. In the future, we expect to see more interesting work based on TRAC, such as better solvers with mechanisms to learn both preconditions and effects, and novel generalization tests that call for more specific reasoning abilities.

Acknowledgement

We acknowledge support from the Natural Science Foundation of China under Grant No. 62076261 and 61906216.

References

Gabor Angeli, Neha Nayak, and Christopher D. Manning. 2016. Combining natural logic and shallow reasoning for question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Antoine Bosselut, Corin Ennis, Omer Levy, Ari Holtzman, Dieter Fox, and Yejin Choi. 2018. Simulating action dynamics with neural process networks. In *International Conference on Learning Representations*.

Ronald J. Brachman and Hector J. Levesque. 2004. *Knowledge Representation and Reasoning*. Elsevier.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2020. Transformers as soft reasoners over language. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3882–3890. International Joint Conferences on Artificial Intelligence Organization. Main track.

Stephen A. Cook and Yongmei Liu. 2002. A complete axiomatization for blocks world. In *International Symposium on Artificial Intelligence and Mathematics, AI&M 2002, Fort Lauderdale, Florida, USA, January 2-4, 2002*.

Bhavana Dalvi, Lifu Huang, Niket Tandon, Wen-tau Yih, and Peter Clark. 2018. Tracking state changes in procedural text: a challenge dataset and models for process paragraph comprehension. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1595–1604. Association for Computational Linguistics.

Soham Dan, Michael Zhou, and Dan Roth. 2021. Generalization in instruction following systems. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 976–981. Association for Computational Linguistics.

Richard Fikes and Nils J. Nilsson. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artif. Intell.*, 2(3/4):189–208.

George Lakoff. 1970. Linguistics and natural logic. *Synthese*, 22(1):151–271.

Hector J. Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR 2012, Rome, Italy, June 10-14, 2012*. AAAI Press.

Belinda Z. Li, Maxwell I. Nye, and Jacob Andreas. 2021. Implicit representations of meaning in neural language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 1813–1827. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

John McCarthy. 1963. Situations, actions, and causal laws. Reprinted in Minsky69Book, pages 410–418.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 784–789. Association for Computational Linguistics.

Raymond Reiter. 2001. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. The MIT Press.

Swarnadeep Saha, Sayan Ghosh, Shashank Srivastava, and Mohit Bansal. 2020. Prover: Proof generation for interpretable reasoning over rules. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 122–136. Association for Computational Linguistics.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 3261–3275.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomás Mikolov. 2016. Towards ai-complete question answering: A set of prerequisite toy tasks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Danfei Xu, Roberto Martín-Martín, De-An Huang, Yuke Zhu, Silvio Savarese, and Li Fei-Fei. 2019. Regression planning networks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 1317–1327.

Rowan Zellers, Ari Holtzman, Matthew E. Peters, Roozbeh Mottaghi, Aniruddha Kembhavi, Ali Farhadi, and Yejin Choi. 2021. Piglet: Language grounding through neuro-symbolic interaction in a 3d world. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 2040–2050. Association for Computational Linguistics.

Shuyan Zhou, Pengcheng Yin, and Graham Neubig. 2021. Hierarchical control of situated agents through natural language. *CoRR*, abs/2109.08214.

A Experiment Details

A.1 GEs for RoBERTa-large, GPT-2, and T5

The results for GEs for RoBERTa-large, GPT-2, and T5 can be seen from Table 4, Table 5, and Table 6 respectively, which suggests:

- Number of parameters matters. Compare the results of RoBERTa-base and RoBERTa-large, we can clearly see that the larger models outperform the smaller ones consistently. However, training larger LMs is notoriously more time-consuming and expensive. Additionally, larger models hinder practical applications in the real world where inference time is typically short.
- The overall generalization performances are similar for different transformer architectures. As the parameters of models (RoBERTa-base, GPT-2 and T5) are at the same level, their struggle facing the generalization tests are alike.

The data efficiency test for GE2 shows that transformers plateau once given more-than-abundant training examples.

A.2 Computing Infrastructure

We use a workstation with Intel i9-10980XE CPU, 128GiB of RAM, and RTX 3090 GPU. The experiments took about 100 hours.

A.3 License of TRAC

The datasets and the code are released under the CRAPL license (the Community Research and Academic Programming License). The full text of the license is included in the supplmentary material.

A.4 Transformers

We use the HuggingFace `transformers` (Wolf et al., 2020) implementation. The following shows the LMs with the number of parameters:

- RoBERTa-base: 125M;
- GPT-2-small: 117M;
- T5-base: 220M;
- RoBERTa-large: 770M.

The following hyper-parameters are used:

- The learning rate is 1e-5 for RoBERTa-base and 1e-4 for the others;
- The maximum sequence length is 256;
- The batch size is 16;
- The weight decay is 0.01;
- The warmup ratio is 0.06.

We repeated every experiment five times to calculate the mean values and standard deviations using five different seeds.

We list the input-output examples in Listing 3.

B The Blocks World

B.1 Symbolic Forms

We list the symbolic forms of both the BW domain and the TRAC examples in the paper.

- Symbolic forms of the problems in Table 1 are shown in Listing 1.
- The action domain BW is defined in a PDDL file. Its content is shown in Listing 2.

Listing 1 TRAC Symbolic Examples

```
Projection Example
Initial state = {
  onTable(Green),
  clear(Red),
  clear(Blue),
  clear(Green),
  onTable(Red),
  onTable(Blue)
}
Action sequence = [
    moveFromTable(Green, Red)
]
Query = on(Blue, Red)

Executability Example
Initial state = {
  onTable(Olive),
  on(Yellow, Olive),
  clear(Indigo),
  on(Indigo, Yellow)
}
Query = [
    moveToTable(Indigo, Yellow)
]

Plan-Verification Example
Initial state = {
  clear(Blue),
  on(Blue, Magenta),
  on(Magenta, White),
  onTable(White)
}
Goal = !on(Blue, Magenta)
Query = [
    moveToTable(Blue, Magenta)
]

Goal-Recognition Example
Initial state = {
  clear(Blue),
  on(Blue, Magenta),
  on(Magenta, White),
  onTable(White)
}
Action Sequence = [
    moveToTable(Blue, Magenta)
]
Query = on(Blue, Magenta)
```

Task	**SD**	**GE1**	**GE2**		**GE3**	**GE4**	
			L4	**L5**		**Literals**	**Conj.**
PR	98.79 (0.0172)	57.81 (0.0238)	94.25 (0.0583)	88.16 (0.0830)	97.60 (0.0228)	100.00 (0.0000)	74.41 (0.0363)
EX	99.85 (0.0007)	96.19 (0.0128)	89.97 (0.0132)	86.33 (0.0187)	99.48 (0.0011)	N/A	N/A
PV	93.94 (0.0099)	84.69 (0.0846)	63.35 (0.0270)	56.69 (0.0215)	96.61 (0.0051)	98.65 (0.0037)	72.84 (0.0839)
GR	98.70 (0.0014)	88.65 (0.0170)	N/A	N/A	97.65 (0.0044)	100.00 (0.0000)	74.07 (0.0050)

Table 4: Accuracies and standard deviations of the RoBERTa-large models on generalization experiments. (SD=Standard Dataset, PR=Projection, EX=Executability, PV=Plan-Verification, GR=Goal-Recognition)

Task	**SD**	**GE1**	**GE2**		**GE3**	**GE4**	
			L4	**L5**		**Literals**	**Conj.**
PR	85.13 (0.0336)	68.73 (0.0397)	70.75 (0.0443)	69.73 (0.0447)	83.79 (0.0257)	89.08 (0.0524)	66.49 (0.0280)
EX	99.37 (0.0037)	90.95 (0.0350)	89.84 (0.0331)	88.11 (0.0303)	97.40 (0.0103)	N/A	N/A
PV	90.09 (0.0157)	79.45 (0.0359)	61.93 (0.0229)	57.13 (0.0172)	91.97 (0.0171)	96.75 (0.0238)	62.83 (0.0194)
GR	97.44 (0.0021)	91.12 (0.0121)	N/A	N/A	94.84 (0.0074)	100.00 (0.0000)	73.47 (0.0099)

Table 5: Accuracies and standard deviations of the GPT-2 models on generalization experiments. (SD=Standard Dataset, PR=Projection, EX=Executability, PV=Plan-Verification, GR=Goal-Recognition)

Task	**SD**	**GE1**	**GE2**		**GE3**	**GE4**	
			L4	**L5**		**Literals**	**Conj.**
PR	82.99 (0.0227)	68.35 (0.0132)	67.69 (0.0188)	66.80 (0.0163)	81.34 (0.0230)	83.27 (0.0237)	71.35 (0.0082)
EX	98.83 (0.0024)	89.36 (0.0308)	81.77 (0.0222)	80.15 (0.0155)	98.12 (0.0014)	N/A	N/A
PV	87.73 (0.0110)	81.03 (0.0588)	68.25 (0.0127)	61.92 (0.0142)	89.49 (0.0137)	97.72 (0.0038)	65.68 (0.0286)
GR	94.04 (0.0082)	82.74 (0.0366)	N/A	N/A	90.61 (0.0134)	99.95 (0.0005)	81.32 (0.0075)

Table 6: Accuracies and standard deviations of the T5 models on generalization experiments. (SD=Standard Dataset, PR=Projection, EX=Executability, PV=Plan-Verification, GR=Goal-Recognition)

Listing 2 The action domain BW in PDDL

```
(define
  (domain blocksworld)
  (:requirements :strips :typing)
  (:types block - object)
  (:predicates  (clear ?x - block)
          (on ?x - block ?y - block)
          (ontable ?x - block))
  (:action move
    :parameters (?x - block ?y -
        block  ?z - block)
    :precondition (and (clear ?x)
              (clear ?z)
              (on ?x ?y))
    :effect (and (not (clear ?z))
           (not (on ?x ?y))
           (on ?x ?z)
           (clear ?y)))

  (:action movetotable
    :parameters (?x - block ?y -
        block)
    :precondition (and (clear ?x)
              (on ?x ?y))
    :effect (and (not (on ?x ?y))
           (clear ?y)
           (ontable ?x)))

  (:action movefromtable
    :parameters (?x - block ?y -
        block)
    :precondition (and (ontable ?x)
              (clear ?x)
              (clear ?y))
    :effect (and (not (ontable ?x))
           (not (clear ?y))
           (on ?x ?y)))
)
```

Listing 3 Input And Output for Transformers

```
RoBERTa
Input: "<s> The yellow block is on
    the table. The magenta block is
    on top of the pink block. The
    gray block is clear. The gray
    block is on the table. The
    magenta block is clear. The pink
    block is on top of the green
    block. The green block is on the
    table. The yellow block is clear.
     Jane moves the yellow block from
     the table to the gray block. </s
    > The green block is clear. The
    gray block is not on top of the
    yellow block. </s>"
Output: 0

GPT-2
Input: "The yellow block is on the
    table. The magenta block is on
    top of the pink block. The gray
    block is clear. The gray block is
     on the table. The magenta block
    is clear. The pink block is on
    top of the green block. The green
     block is on the table. The
    yellow block is clear. Jane moves
     the yellow block from the table
    to the gray block. The green
    block is clear. The gray block is
     not on top of the yellow block."
Output: 0

T5
Input: (Same as GPT-2)
Output: "No"
```

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
6

□ A2. Did you discuss any potential risks of your work?
Not applicable. Left blank.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Left blank.

□ B1. Did you cite the creators of artifacts you used?
Not applicable. Left blank.

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Left blank.

□ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Not applicable. Left blank.

□ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. Left blank.

□ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Not applicable. Left blank.

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
3

C ☑ Did you run computational experiments?

4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
A

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
A

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
4

□ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Not applicable. Left blank.

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?
Left blank.

□ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

□ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

□ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

□ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

□ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.