Testing the limits of logical reasoning in neural and hybrid models

Manuel Vargas Guzmán
University of Warsaw
m.vargas-guzman@uw.edu.pl

Jakub Szymanik
University of Trento
jakub.szymanik@gmail.com

Maciej Malicki
Institute of Mathematics of the Polish Academy of Sciences
mmalicki@impan.pl

Abstract

We study the ability of neural and hybrid models to generalize logical reasoning patterns. We created a series of tests for analyzing various aspects of generalization in the context of language and reasoning, focusing on compositionality and recursiveness. We used them to study the syllogistic logic in hybrid models, where the network assists in premise selection. We analyzed feed-forward, recurrent, convolutional, and transformer architectures. Our experiments demonstrate that even though the models can capture elementary aspects of the meaning of logical terms, they learn to generalize logical reasoning only to a limited degree.

1 Introduction

Despite the enormous successes of models based on deep learning, we still need to know more about how and what these models learn. The question of fundamental importance is to what extent they can 'grasp' the rules (or – more generally – the structure) governing involved data and tasks. It can be phrased as the problem of generalization, i.e., the ability to perform on data unseen during training.

Language structure is well understood from several perspectives: grammar, semantics, or rules of reasoning have been extensively studied and successfully formalized. However, even in this area, despite the available theoretical background, the methodology for studying generalization is still not well developed. The need for a systematic approach to this problem is indicated by a recent survey (Hupkes et al., 2023) of generalization research in NLP.

So far, the study of neural models for tasks related to logic and reasoning is rather limited. An early attempt is Bowman et al. (2015), where networks learn logical relations, such as entailment, between pairs of sentences in a simple artificial language. More recent work Ontanon et al. (2022) involves models that determine whether a given inference can be proved from a given set of premises by providing the list of inference rules as an output. In Clark et al. (2021), models learn to reason with prescribed rules, while in Schlegel et al. (2022), the authors consider models deciding whether a given set of sentences is consistent. It is worth mentioning that investigating reasoning has a sound linguistic motivation. To take a straightforward example, it is hard to argue that a model grasps the meaning of quantifier "all" if it is not able to perform reasonings of the form: "All a are b" and "All b are c" implies "All a are c."

a)

$$\begin{tikzcd}
c & & d \arrow[rr, leftrightarrow, "E"] & & f \arrow[d, bend left, "O"] \\
& b \arrow[ul, "A"] \arrow[ur, "A"'] & & & e \arrow[u, "A"] \\
& a \arrow[u, "A"] & & &
\end{tikzcd}$$

b)
$$\begin{array}{ll} X_1 = & [Aab, Abc, Abd, Aef, Edf, Ofe, Eae] \\ Y_1 = & [\ 1 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0\ ] \end{array}$$

c)
$$\begin{array}{ll} X_2 = & [Aab, Abc, Abd, Aef, Edf, Ofe, Eab] \\ Y_2 = & [\ 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0\ ] \end{array}$$

Figure 1: a) Example of a simple knowledge base $\mathcal{KB} = \{Aab, Abc, Abd, Aef, Edf, Ofe\}$ b) Example of input X_1 and label Y_1 to build the inference $\{Aab, Abd, Aef, Edf\} \vdash Eae$. The input always contains the whole knowledge base $\mathcal{KB}$ and a hypothesis H at the end. Formulas are encoded as 1-hot vectors; the label is a binary vector indicating the premises needed to derive H, if it is valid, or the 0-vector, otherwise. c) Example of input X_2 and label Y_2 for invalid hypothesis Eab.

In this paper, we focus on logical reasoning in the syllogistic fragment of the natural language. The syllogistic logic has nice properties, e.g., soundness and completeness. Notably, the logic is non-trivial but still sufficiently elementary to play the role of a benchmark for models of reasoning.

We investigate the generalization of inference patterns in the training data and the following task. The network, presented with a knowledge base $\mathcal{KB}$ (i.e., a set of premises) and a hypothesis H selects the premises required to construct a proof of H from $\mathcal{KB}$ (if it exists); see Figure 1 for an illustrative example. Thus, one can think of our models as hybrid models: by selecting premises, the network assists the prover that is supposed to construct a proof. The paper can also be described as a study of reasoning in the presence of multiple premises, a research line rarely explored in deep learning.

We are mainly interested in two aspects of generalization: *recursiveness* (elements can be iteratively combined) and *compositionality* (structures are determined by their constituents). It is worth emphasizing that they are frequently conflated even though conceptually different. There are fully recursive systems that are not compositional, the best-known example being Tarski's interpretation of first-order logic; see Janssen and Partee (1997) for a detailed discussion and more examples. There are also fully compositional structures with limited recursiveness, e.g., Boolean operations on a finite family of sets are compositional but can be combined only in a finite number of ways.

In the context of reasoning, we will say that a model processes inferences in a recursive manner if it is capable of applying inference patterns learned during training to more complex instances. Going back to the previous example, if the model knows that "All a are b" and "All b are c" implies "All a are c," it should also be able to conclude from the extra piece of information "All c are d" that "All a are d." Compositionality means the converse situation: provided that the model knows how to apply an inference pattern to complex instances, it should be able to do so for simpler ones. In other words, the derivation of "All a are d" from "All a are b," "All b are c," and "All c are d," should be accompanied by the derivation of "All a are c."

In the study, we employed different types of architectures, Multilayer Perceptron, Recurrent Neural Networks, Convolutional Neural Networks, and Transformers, to compare their performance and capabilities for generalization on artificially generated syllogistic corpora. On the surface of things, the models manage to learn the assigned task almost perfectly (see Table 3); in particular, the generalization gap, which is a standard measure of generalization, is very low. However, the experiments designed to verify recursive and compositional generalization reveal that neural networks—and the hybrid models they comprise—poorly generalize, regardless of architecture. In particular, this sheds light on the purported superiority of the transformer architecture. On the positive side, some evidence for recursive generalization can be observed.

Last but not least, one of our primary goals is to contribute to developing a methodology for investigating generalization in the context of recursiveness and compositionality. The approach proposed in this paper can be exploited in other settings, either directly related to reasoning, e.g., other fragments of language and inference systems, or not, e.g., sequence-to-sequence models studied in Hupkes et al. (2019) and Lake and Baroni (2023).

2 Syllogistic Logic

Pratt-Hartmann Pratt-Hartmann (2004) defines a *fragment* of a natural language as a subset of that language with an uncontroversial translation into a formal language that reconstructs logical entailment. The syllogistic fragment, first introduced and studied by Aristotle, is the simplest non-trivial language fragment. Aristotle considered only syllogisms consisting of two premises and a conclusion. A well-known example is "If all men are mortal and all Greeks are men, then all Greeks are mortal." However, classical syllogistic can be easily extended to inferences involving more than two premises, see, e.g., Łukasiewicz (1951); Smiley (1973). In our setting, only general names with non-empty denotations are allowed. Thus, "Socrates is a man" is not a syllogistic formula for us, while "Every unicorn is an animal" implies "Some unicorn is an animal".

2.1 Language

The syllogistic comprises the formulas Aab ("Every a is b"), Eab ("No a is b"), Iab ("Some a is b"), and Oab ("Some a is not b"). The former two are called *universal* formulas since the translation to the first order logic is $\forall x.[A(x) \to B(x)]$ and $\forall x.[A(x) \to \neg B(x)]$, respectively. And the latter two are *existential* formulas represented as $\exists x.[A(x) \wedge B(x)]$ and $\exists x.[A(x) \wedge \neg B(x)]$, respectively. Note that translations of Aab and Oab are contradictory, and so are Iab and Eae. Moreover, existential formulas are symmetric, *i.e.*, Iab and

Iba have equivalent translations, and so do Eab and Eba.

We define a language as follows: let $\mathcal{V} = (Q, \mathcal{C})$ be a vocabulary of quantifier symbols $Q = \{A, E, I, O\}$ and constant symbols $\mathcal{C} = \{a, b, c, \ldots\}$. Formulas are built as Axy, Exy, Ixy, or Oxy, where $x, y \in \mathcal{C}$, $x \neq y$. In particular, Aaa is not a formula.

There is no negation in our language; however, we denote the "contradiction" of a formula F by $\overline{F}$, *i.e.*, $\overline{Aab} = Oab$, $\overline{Oab} = Aab$, $\overline{Iab} = Eab$, and $\overline{Eab} = Iab$.

An *A-chain*, denoted as $Aa - b$, represents either the formula Aab or the sequence of two or more formulas $Aac_1, Ac_1c_2, \ldots, Ac_{n-1}c_n, Ac_nb$ (for $n \geq 1$). Finally, a *knowledge base* is a finite set of formulas or *premises*.

2.2 Types of syllogistic inferences

In this paper, we follow Smiley (1973). However, we do not delve into details; in particular, we do not specify the proof system because it does not matter in our framework. The aforementioned translation of syllogistic formulas into first-order logic allows for interpreting formulas by interpreting constants as non-empty unary predicates. This is sufficient to define the notions of consistency and inference. A set $\mathcal{F}$ of formulas is *consistent* if there is an interpretation of constants that makes all formulas in $\mathcal{F}$ true. A formula F is a conclusion from a set of premises $\mathcal{F}$ if $\mathcal{F} \cup \{\overline{F}\}$ is inconsistent. We write $\mathcal{F} \vdash F$ for the *inference* formed by premises $\mathcal{F}$ and conclusion F. Given a knowledge base $\mathcal{KB}$, a hypothesis H is *valid* if $\mathcal{KB} \vdash H$, otherwise H is *invalid*.

In the paper, we are interested in minimal inferences, i.e., inferences $\mathcal{F} \vdash F$ such that $\mathcal{F}' \nvdash F$ for any proper subset $\mathcal{F}' \subset \mathcal{F}$. For example, $\{Abc, Abd\} \vdash Icd$ is minimal, while $\{Aab, Abc, Abd\} \vdash Icd$ is not because Aab is not needed to infer the conclusion. Minimal inferences correspond to *antilogisms*, i.e., minimal inconsistent sets of syllogistic formulas.

Theorem 1 (Smiley (1973)). *Every antilogism is of the following form* $\{Aa - b, Oab\}$, $\{Aa - b, Aa - c, Ebc\}$, *or* $\{Aa - b, Ac - d, Iac$ *(or* Ica*)*$, Ebd\}$.

Theorem 2 (Smiley (1973)). *Let F be a formula and $\mathcal{F}$ be a set of formulas. $\mathcal{F} \cup \{\overline{F}\}$ is an antilogism if and only if $\mathcal{F} \vdash F$, and $\mathcal{F} \vdash F$ is minimal.*

All minimal syllogistic inference types can be easily recovered from the above theorems. The final list is presented in Table 1 (see A.1 for more details). To cover all syllogisms, symmetric formulas need to be used interchangeably, *e.g.*, $Ixy = Iyx$; formulas of the form Aaa are disregarded.

(1) $\{Aa - b, Ac - d, Oad\} \vdash Obc$
(2) $\{Aa - b\} \vdash Aab$
(3) $\{Aa - b, Ac - d, Aa - e, Ede\} \vdash Obc$
(4) $\{Aa - b, Aa - c\} \vdash Ibc$
(5) $\{Aa - b, Ac - d, Ae - f, Iae, Edf\} \vdash Obc$
(6) $\{Aa - b, Ac - d, Ebd\} \vdash Eac$
(7) $\{Aa - b, Ac - d, Iac\} \vdash Ibd$

Table 1: List of all types of syllogistic inferences

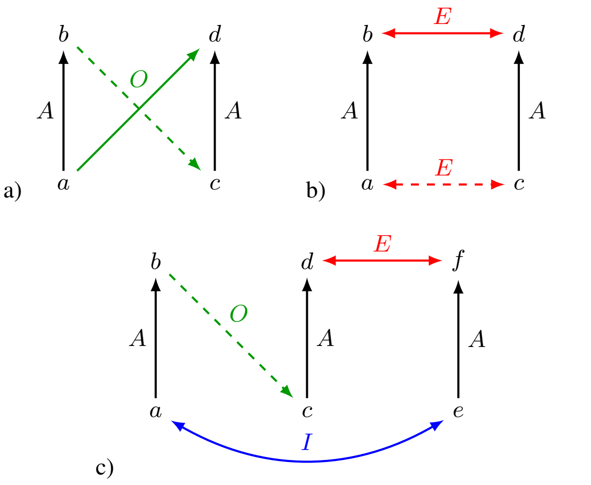

Figure 2: Diagrams illustrating examples of types of syllogistic inferences (dashed lines represent conclusions) a) Type (1) $\{Aa - b, Ac - d, Oad\} \vdash Obc$ b) Type (6) $\{Aa - b, Ac - d, Ebd\} \vdash Eac$ c) Type (5) $\{Aa - b, Ac - d, Ae - f, Iae, Edf\} \vdash Obc$

To give the reader a better idea of what syllogistic inferences look like, we present in Figure 2 diagrams illustrating some of them.

We say that an inference $\mathcal{F} \vdash F$ can be decomposed into inferences $\mathcal{F}_1 \vdash F_1$, $\mathcal{F}_2 \cup \{F_1\} \vdash F$, if $\mathcal{F} = \mathcal{F}_1 \dot{\cup} \mathcal{F}_2$, i.e., the premises can be split into two disjoint subsets $\mathcal{F}_1$ and $\mathcal{F}_2$ so that $\mathcal{F}_1$ forms premises of the first inference, and $\mathcal{F}_2$, together with the conclusion F_1 from $\mathcal{F}_1$, forms the premises of the second one. The main observation here is that every inference can be decomposed into an inference with all A-chains of length 1 and an inference of type 2 (see Table 1). Other decompositions, discussed in 4.2, are also possible for some inference types.

3 Synthetic Data and Neural Models

In order to avoid problems related to the choice of premises needed to infer a given hypothesis, we used only *non-redundant* knowledge bases, i.e., knowledge bases such that for every valid hypothesis, there is a unique minimal set of premises that proves it.

We represent a knowledge base as a graph where vertices are constants and edges denote quantifiers (see Figure 1a). All A-formulas are a set of m disjoint trees $T_1, \ldots, T_m$ (or a *forest*). Each tree is a directed graph $T_i = (V, E)$ such that there is at most one path between any two vertices. We created synthetic consistent non-redundant knowledge bases $\mathcal{KB}$ for training and testing the neural models using the following general algorithm:

1. Randomly generate a forest where each vertice corresponds to a constant and every directed path between two vertices corresponds to an A-chain.
2. For every pair of (different) trees (T_i, T_j):

   Add one E-formula and one I-formula between T_i and T_j.
3. For each tree T_i:

   Add O-formulas within T_i.

We randomly add formulas (steps 2. and 3.) such that there is no redundancy and the set $\mathcal{KB}$ remains consistent.

For every experiment we generated a consistent non-redundant knowledge base $\mathcal{KB} = \{P_1, \ldots P_n\}$ of n premises. We trained neural models using a *multi-label* approach and supervised learning techniques. Each element of the dataset consists of an input X associated with a label Y. The input vector X encodes the knowledge base $\mathcal{KB}$ and a hypothesis H. For a valid H, the label Y is a binary vector of size n that tags all the necessary premises to derive H by assigning 1 to every Y_i if $\mathcal{KB} \setminus \{P_i\} \nvdash H$ and 0, otherwise. For invalid H, Y is the zero vector.

We stratify the training/test split by types of inferences, for every valid type we train 75% and test on the remaining 25%. For invalid hypotheses, we only train 20%, since they make up more than 80% of the data (see Table 2 for distribution of hypotheses). In some experiments, the stratification somewhat differs (i.e., when we remove an inference type from the training data), but, in general, we stick to the above stratification principles.

We used *one-hot encodings* to produce input vectors. Each constant and quantifier are represented as a one-hot vector of dimension d (where d is the size of the vocabulary). We also tested word embeddings to encode knowledge bases, but one-hot encodings give better performance (see A.2).

4 Experiments and Results

The data and scripts to run these experiments are available online[1]. We randomly generated 5 consistent knowledge bases. Each of them consists of 4 trees, 66 constants, and 78 formulas. We made sure that no valid hypothesis gave rise to the same label in two different knowledge bases. In the following experiments, we trained 4 different architectures of neural models: *Multilayer Perceptron* (MLP), *Recurrent Neural Network* (RNN), *Convolutional Neural Network* (CNN), and *Transformers* (TRA) (as a matter of fact, we also considered some variants of these architectures, e.g., LSTM or GRU, but the results were very similar). We employed grid-search techniques to optimize the configurations for overall performance. The detailed description of optimization procedures and final specifications can be found in A.3. We trained each knowledge base for 3 runs.

Being part of a hybrid model, networks are supposed to provide premises for the prover. Therefore, beside the standard measure of accuracy (correct label), we consider another one: an output is correct if it involves all the necessary premises, i.e., it is a correct but not necessarily minimal (NNM) inference.

4.1 Overall Performance

In the first experiment, we checked the overall accuracy of the models for the split described in 3. The results are shown in Table 3 (more details in A.4). Clearly, the numbers are high enough to exclude a large generalization gap (see, e.g., Hoffer et al. (2017)), i.e., a substantial difference in performance on the training and on the test data (see A.5 for exact values). A large generalization gap would indicate that the model excessively memorizes (overfits) training data. However, as the next experiments show, the generalization gap is not a good measure of compositional and recursive generalization.

We also verified how the models generalize basic non-compositional and non-recursive features

[1]https://github.com/manuel-vg/syllogistic-logic

Type	1	2	3	4	5	6	7	Valid	Invalid	All
# Inf.	124	334	519	1026	157	245	622	3027	14133	17160

Table 2: Data distribution of the 5 knowledge bases used for training (mean # of inferences by type and validity)

Model	Inf.	Best	Mean	SD	NNM
MLP	Val.	93.9	83.2	13.1	88.9
	Inv.	97.1	94.2	2.5	–
	All	96.6	93.5	3.1	–
RNN	Val.	95.9	93.5	1.3	95.3
	Inv.	98.3	97.7	0.5	–
	All	98.0	97.4	0.4	–
CNN	Val.	94.3	92.0	1.3	94.4
	Inv.	97.3	96.7	0.3	–
	All	96.9	96.4	0.2	–
TRA	Val.	96.6	93.6	2.9	95.7
	Inv.	97.8	96.3	1.3	–
	All	97.7	96.1	1.3	–

Table 3: Overall accuracy: best, mean, standard deviation (SD), mean accuracy for not-necessarily-minimal correct inferences (NNM), for valid (Val.), invalid (Inv.), and all hypotheses, respectively (see 4.1)

of the syllogistic logic: Principle of Contradiction (either H or $\overline{H}$ is invalid), non-empty denotations of constants (if Aab is valid, then Iab is valid), as well as the symmetry of formulas Iab and Eab. The level of generalization is very high (see A.7). It suggests that the models learned at least elementary aspects of the meaning of involved terms (see Discussion).

4.2 Compositionality

Unseen Short Lengths. We define the *length* of inference as the total length of all A-chains, *i.e.*, the number of A-formulas among the premises. To perform the unseen lengths experiments, for the training data, we removed inferences either with short or with long lengths, the length depending on inference type (this is because maximal lengths $\mu(t)$ represented in the knowledge base depend on inference type t). Then we test only on the eliminated inferences.

In this experiment, we removed inferences of length 5 and less. Accuracies calculated for every unseen length separately are shown in Figure 3 (the left plot). A sharp and consistent drop in performance can be observed, depending on how far the tested length is from the lengths present in the training data.

We interpret these results as a clear sign of a lack of compositionality. The models are able to perform well on the longer inferences without being able to perform on shorter ones, even though the latter form parts of the former. To take a simple inference of type 2 as an example, if the model is able to conclude from Aab, Abc, Acd that Aad but not that Aac, it means that this inference is not compositional.

Removing an inference type. For this experiment, we proceeded to split the training/test dataset in a way similar to that described in 3, the only difference being that an entire type of inference is removed from the training dataset. We then checked the performance by testing on each type separately. Table 4 presents the results (mean accuracy) of tests on the removed type, which are most relevant from our perspective.

The first observation is that the categorization of the data based on inference types is not spurious. The models are essentially incapable of finding inferences of types that are not present in the training data. On the other hand, these results confirm our conclusion from the experiment on short unseen lengths: the models do not use compositional inferences.

Compositional inferences presuppose recognizing the inferential structures of its parts. It has been noted in 2.2 that an inference of every type can be decomposed into two inferences, one of which is of type 2. There are other possible decompositions. For example, an inference of type 5 requires knowledge that Iea and $Aa - b$ imply Ieb, i.e., it can be decomposed into two inferences, one of which is of type 7. There are similar relationships between type 5 and type 6, or type 3 and types 6 and 7. Therefore removing a type from the training data would not completely annihilate performance on this type for a model that processes inferences in a compositional manner.

The only exception is type 3, on which all the architectures exhibit non-zero performance after removing it from the training data. However, this can be explained by the models' grasping the non-empty denotations of constants (i.e., that Aab implies Iab). With the aid of this generalization, type 3 can be derived from type 5. Indeed, after remov-

Model	1	2	3	4	5	6	7
MLP	0.0	0.0	7.1	0.1	0.0	0.0	0.0
RNN	0.0	0.0	18.5	0.6	0.0	0.0	0.0
CNN	0.0	0.0	7.0	0.3	0.0	0.0	0.0
TRA	0.0	0.0	13.2	2.5	0.1	0.0	0.0

Table 4: Mean accuracy for testing on a type that was removed from the training data

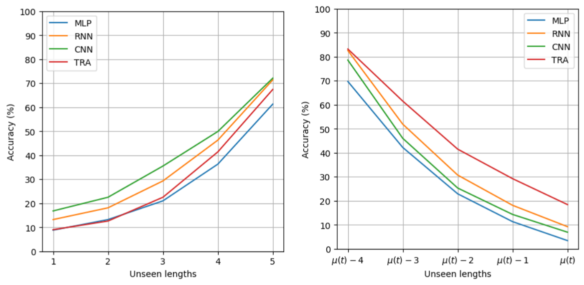

Figure 3: Performance on unseen lengths for short inferences (left) and long inferences (right). The models are trained on inferences of length more than 5 (left) or less than $\mu(t) - 4$ (right), where $\mu(t)$ is the maximal length for type t. Then they are tested on the lengths removed from training. The plots show accuracies for each removed length separately.

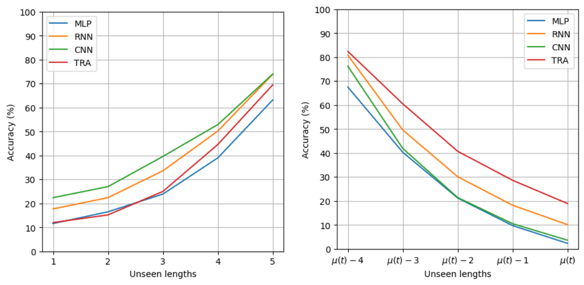

Figure 4: Unseen lengths for short inferences (left) and long inferences (right) without type 2 for test.

ing additional type 5, the performance of type 3 drops to zero.

4.3 Recursiveness

Unseen Long Lengths. This experiment is similar to the experiment on unseen short lengths but with the longest inferences removed from the training data. The results for inferences of length more than $\mu(t) - 5$ removed (i.e., the 5 longest lengths for each type), and accuracies calculated for every unseen length separately are shown in Figure 3. Again, we can see a very clear drop in performance, depending on the distance of the length from the lengths seen in training. It means that the models are not able to perform inferences much longer than those used for training. As a matter of fact,

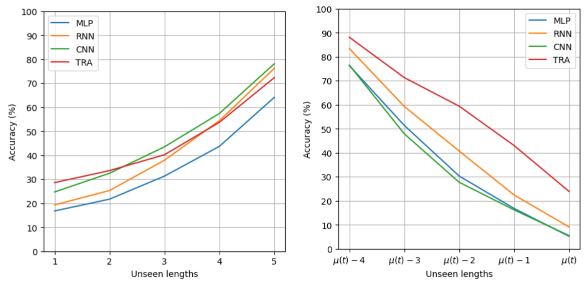

Figure 5: Performance on unseen lengths for short inferences (left) and long inferences (right), except for type 2. The experiments are as in Figure 4, but with all the lengths for type 2 inferences used in training (see 4.3 for details).

for inferences longer only by 1, the accuracy is still high.

Clearly, every inference type has a recursive structure: longer inferences can be constructed from shorter ones by extending the involved A-chains. This kind of recursiveness, consisting of the iterative application of a rule, is termed in Hupkes et al. (2019) as *productivity*. Thus, we can interpret the results of this experiment as a sign of a lack of productivity. On the other hand, the results, when only inferences of maximal length are removed, indicate that some local extrapolation takes place.

Unseen Lengths except for type 2. In these two experiments, we removed from the training data either the shortest or the longest inferences, except for inferences of type 2. These are selected without any restrictions on the length (but not included in the test data). The results are presented in Figure 5.

The performance drops, but the change is smaller as compared to the experiments on unseen lengths described above and in 4.2 (see Figure 4 for comparison). This is particularly evident for TRAs, e.g., for short unseen lengths 1,2,3, the difference is 16.6, 18.4 and 15.2, respectively. For long unseen lengths, the corresponding values are 5, 14.4 and 18.7. Interestingly, RNNs and CNNs do not seem to considerably benefit from extra training

Model	Best	Mean	SD
MLP	0.0	0.0	0.0
RNN	0.0	0.0	0.0
CNN	0.0	0.0	0.0
TRA	0.0	0.0	0.0

Table 5: Unseen combinations of premises

data. A more detailed discussion of the general performance of the architectures will be carried out in a separate section.

We interpret these results as a sign of some capabilities of the models to combine inferences, i.e., as evidence for some level of recursiveness. As it was pointed out in the introduction, compositionality and recursiveness are distinct categories of language and language processing, so our findings from this and the previous section, indicating a lack of compositionality and some presence of recursiveness, are not contradictory.

Unseen combinations of premises. In this experiment, we select a set Δ of formulas forming an A-chain from the knowledge base, remove from the training data inferences $\mathcal{F} \vdash F$ such that

$$|\mathcal{F} \cap \Delta| > 1,$$

and test on the removed inferences. In other words, during training, the models do not see inferences that combine two or more premises from Δ. This aspect of generalization is termed *systematicity* in Hupkes et al. (2019).

For $n \in \{2, 4, 6, 8\}$, we randomly selected an A-chain Δ of length n, and performed the experiment. The results presented in Table 5 (mean for all values of n) are rather extreme: all architectures exhibited zero accuracy. Apparently, in order for the models to be able to employ a combination of premises in an inference, the premises need to be seen together in some inference during training. It is true even of the simplest inferences like $\{Aab, Abc\} \vdash Aac$.

4.4 Testing on a new knowledge base

In the last experiments, we went beyond the general framework of the study. We substantially increased the distance between the training and the test data, employing a new knowledge base for testing. We selected 3 bases with no overlapping labels for a given hypothesis and repeated the experiment 6 times for every combination of the base used for training and for testing.

Model	Inf.	Best	Mean	SD
MLP	Val.	0.1	0.0	0.0
	Inv.	67.5	63.3	2.3
	All	55.2	51.8	1.9
RNN	Val.	0.0	0.0	0.0
	Inv.	30.1	11.2	6.6
	All	24.5	9.2	5.4
CNN	Val.	0.0	0.0	0.0
	Inv.	100.0	95.3	6.6
	All	81.7	78.0	5.4
TRA	Val.	0.0	0.0	0.0
	Inv.	83.1	81.9	0.7
	All	67.9	67.1	0.6

Table 6: Overall accuracy for tests on new knowledge bases

The results in Table 6 show that in this setting, the models generalize poorly. In particular, the accuracy on valid hypotheses is always zero. A more detailed analysis of the results reveals (see A.6) that some architectures learn to ignore the knowledge base part of the input and, regardless of the test data, produce labels that correspond to the base used for training. This is true of TRAs, and, to a lesser extent, of CNNs and MLPs. However, RNNs do not memorize in this way.

On the other hand, CNNs exhibit almost perfect performance (mean: 95.3%) on invalid hypotheses, and this behavior cannot be explained by memorization: 16% (i.e., around 2300) of the hypotheses that are invalid in the new knowledge base are valid in the old one (see Table 7). The task of deciding if a hypothesis H is invalid for a knowledge base $\mathcal{KB}$ amounts to deciding if the set $\mathcal{KB} \cup \{\overline{H}\}$ is consistent. Thus, CNNs learned to recognize consistency of sets of syllogistic formulas far beyond the training setup. All other architectures obtained zero accuracy on this task.

Finally, we tested the generalization of basic features of the syllogistic logic as in 4.1. The results show almost perfect performance for CNNs and TRAs (see A.7 for details). Interestingly, RNNs exhibit a low level of generalization of the Principle of Contradiction.

4.5 Comparison of architectures

TRAs do not substantially outperform other architectures. This stands in contrast to presuppositions (see, e.g., Smolensky et al. (2022)) that it is transformers' ability to process data in a compositional manner that explains their successes in real-world applications. They do perform better on tests related to recursiveness but are below average on our compositionality tests. More importantly, we never see qualitative superiority, e.g., tasks on which only TRAs attain non-zero performance.

RNNs struggle when tested on a new knowledge base. It is the only architecture that does not generalize the Principle of Contradiction. Moreover, RNNs' limited memorization indicates that they process data differently. CNNs' almost perfect performance on invalid hypotheses hints that they may have some interesting distinctive features deserving of further studies.

MLPs, unsurprisingly, lag behind, but they are not so much worse. Thus, if understood as a benchmark architecture, their performance indicates that in terms of capabilities for compositional and recursive aspects of language processing, all the known deep-learning designs are basically on par – at least when employing standard training regimes.

5 Discussion

The paper's main contributions are two-fold: methodological and experimental.

Studies of logical reasoning in neural networks usually consider much simpler toy logic examples, often not even fully recursive, than the experimental setup offered in this paper, cf. Bowman et al. (2015). On the other hand, articles focusing on various aspects of generalizations, like compositionality, systematicity, or recursiveness, often adopt empirical frameworks less straightforwardly linked to reasoning and semantics, cf. Hupkes et al. (2019) or Lake and Baroni (2023). Moreover, many papers do not distinguish between recursiveness and compositionality. For example, in Lake and Baroni (2023), a sequence-to-sequence model's performance on unseen combinations of functions is tested (see Fig. 2 in the paper); however, it is not verified whether the model can correctly process corresponding sub-combinations, which is a necessary condition for compositionality. Similarly, in Clark et al. (2021), the authors investigate the generalization of certain rule-based reasonings to patterns longer than those seen in training (see Table 1). But they do not take into consideration their internal structure, either.

The current paper proposes solving these problems by a systematic study of reasoning in a natural language fragment Pratt-Hartmann (2004). Our experiments show that even though the neural network models can grasp some elementary aspects of syllogistic reasoning, they cannot learn the logic's fully recursive and compositional nature. They manifest various aspects of the meaning of involved terms, e.g., the Principle of Contradiction, non-emptiness of denotations, or the symmetry of quantifiers. They also exhibit some ability to combine inferences into more complex ones, which agrees with findings, e.g., from Lake and Baroni (2023). At the same time, they do not assimilate the recursive structure of inferences, so high performance on shorter inferences of a given type does not translate to high performance on longer ones (see Schlegel et al. (2022) for similar results for the task of recognizing consistency of a set of formulas). Moreover, the networks do not pass the compositionality test: they appear to apprehend complex inferences without apprehending the constituent subinferences. From the semantical perspective, this shows that the models do not understand the meanings of syllogistic formulas because, ultimately, it is their meanings that determine the structure of syllogistic inferences.

5.1 Acknowledgments

The research was funded by the National Science Center, Poland [grant 2020/37/B/HS1/04220]. We gratefully acknowledge Polish high-performance computing infrastructure PLGrid (HPC Center: ACK Cyfronet AGH) for providing computer facilities and support within computational grant no. PLG/2023/016283. We are also grateful to Raffaella Bernardi, Jaap Jumelet, Paolo Rota, Shane Steinert-Threlkeld, Stefano Teso, Kentaro Yamamoto, the audience of the CLIC seminar at UniTn, and the workshop on reasoning and neural models at UvA for comments and interesting discussions.

5.2 Limitations

The syllogistic form is only a small fragment of the natural language, so our findings are not conclusive with regard to aspects of logical reasoning that are not present in syllogistic logic. Moreover, the choice of encodings and the synthetic data constructed for the sake of experiments conducted in the study further increase the distance of our set-up from natural language reasoning.

Another limitation is related to the training regimes employed. Other methods of training neural networks may allow for a higher level of compositional and recursive generalization.

Model	Inf.	%	Best	Mean	SD
MLP	Val.	16	0.8	0.4	0.2
	Inv.	84	80.0	75.4	2.6
RNN	Val.	16	0.2	0.0	0.0
	Inv.	84	35.7	13.4	7.8
CNN	Val.	16	100.0	88.7	11.5
	Inv.	84	100.0	96.6	6.1
TRA	Val.	16	1.6	0.8	0.3
	Inv.	84	98.4	97.5	0.6

Table 7: Split accuracy for tests on invalid hypotheses in a new knowledge base ($\mathcal{KB}$): 16% (appr. 2500) of hypotheses that are invalid in the new $\mathcal{KB}$ were valid in the $\mathcal{KB}$ used for training.

References

Samuel R. Bowman, Christopher Potts, and Christopher D. Manning. 2015. Recursive neural networks can learn logical semantics. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 12–21, Beijing, China. Association for Computational Linguistics.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2021. Transformers as soft reasoners over language. *IJCAI'20: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, 623(537):3882–3890.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Elad Hoffer, Itay Hubara, and Daniel Soudry. 2017. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. 2019. The compositionality of neural networks: integrating symbolism and connectionism. *CoRR*, abs/1908.08351.

Dieuwke Hupkes, Mario Giulianelli, Verna Dankers, Mikel Artetxe, Yanai Elazar, Tiago Pimentel, Christos Christodoulopoulos, Karim Lasri, Naomi Saphra, Arabella Sinclair, et al. 2023. A taxonomy and review of generalization research in nlp. *Nature Machine Intelligence*, 5(10):1161–1174.

Theo MV Janssen and Barbara H Partee. 1997. Compositionality. In *Handbook of logic and language*, pages 417–473. Elsevier.

Brenden M. Lake and Marco Baroni. 2023. Human-like systematic generalization through a meta-learning neural network. *Nature*, 623(7985):115–121.

Jan Łukasiewicz. 1951. *Aristotle's Syllogistic From the Standpoint of Modern Formal Logic*. Oxford, England: Garland.

Santiago Ontanon, Joshua Ainslie, Vaclav Cvicek, and Zachary Fisher. 2022. Logicinference: A new dataset for teaching logical inference to seq2seq models.

Ian Pratt-Hartmann. 2004. Fragments of language. *Journal of Logic, Language and Information*, 13(2):207–223.

Viktor Schlegel, Kamen Pavlov, and Ian Pratt-Hartmann. 2022. Can transformers reason in fragments of natural language? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11184–11199, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Timothy J. Smiley. 1973. What is a syllogism? *Journal of Philosophical Logic*, 2(1):136–154.

Paul Smolensky, Richard Thomas McCoy, Roland Fernandez, Matthew Goldrick, and Jianfeng Gao. 2022. Neurocompositional computing: From the central paradox of cognition to a new generation of ai systems. *AI Magazine*, 43(3):308–322.

A Appendix

A.1 Construction of inferences

We derived all possible syllogisms from Theorems 1 and 2 as follows: for each antilogism of the form $\mathcal{F} \cup \overline{F}$, we consider all possible values that $\overline{F}$ can have to construct a valid syllogism of the form $\mathcal{F} \vdash F$. Table 18 summarizes this process for every form of antilogism described in Theorem 1. Note that from the third form, *i.e.*, $\{Aa-b, Ac-d, Iac, Ebd\}$ and $\{Aa-b, Ac-d, Ica, Ebd\}$, we only describe the former, since the latter is equivalent but with swapping variables. After renaming variables and removing equivalent syllogisms, the list from Table 18 boils down to 7 types of valid inferences presented in Table 1.

A.2 Types of encodings

We experimented with one-hot and word embeddings to encode syllogistic formulas. We picked the former because it achieves higher accuracy within our framework. To see a comparison between one-hot encoding and word embeddings, we trained a single knowledge base using both techniques, we then tested the overall accuracy for valid hypotheses. The results for each architecture are shown in Figure 8. There is a significant difference in MLPs and TRAs. RNNs did a much better work and CNNs seem to be able to handle both types of

Model	Enc.	Best	Mean	SD
MLP	1-hot	82.8	82.4	0.3
	emb.	65.5	54.9	9.5
RNN	1-hot	86.2	85.6	0.6
	emb.	74.9	74.4	0.4
CNN	1-hot	86.0	85.4	0.8
	emb.	84.4	83.4	0.7
TRA	1-hot	93.7	91.9	2.1
	emb.	65.6	61.0	5.5

Table 8: Comparison between 1-hot encodings and word embeddings using a single knowledge base and the same configuration for each architecture (accuracy for valid hypotheses)

encoding quite well. For word embeddings, increasing the number of heads in the TRA architecture also increases the accuracy. But still, they cannot outperform one-hot representations.

A.3 Neural models specification

We built our models using the *TensorFlow* library and *Python* as a programming language. The gradient descent method we used is the *Adam* optimization algorithm (for MLP, CNN, and TRA) and its variant *Adamax* (for RNN) with a learning rate of 0.001. The number of epochs performed is 350 for transformers and 250 for the rest, and the batch size for all architectures is 20. The configuration of layers used for each model is detailed in Table 9.

We performed our experiments using a *GPU-A100*. The time for training a model varies for each architecture and each experiment. A single run, on average, for MLPs, CNNs, and TRAs takes between 10 and 20 minutes, whereas for RNNs, it takes around 60 minutes.

The number of neurons, layers, and other essential hyperparameters were optimized using grid-searching techniques. Our aim was achieve an optimal performance for the overall accuracy test, in particular for valid inferences. We obtained above 90% of correct predictions for valid and invalid inferences using mostly default parameters and keeping the models with simple and general specifications as much as possible. Nevertheless, we experimented with increasing the number of layers and units or tweaking other parameters such as the learning rate, however without seeing any significant improvements. In particular, adding more layers to RNNs led to the vanishing gradient problem. For CNNs, we also tried different configurations regarding the number of filters, and the sizes of kernels and poolings. Finally, for transformers, we set up an encoder-only model by mainly changing the number of attention layers and attention heads. We chose this type of model since our approach can be seen as a text classification task. However, for completeness, we also experimented with encoder-decoder and decoder-only transformers with unsuccessful results.

We also experimented with LSTM and GRU recurrent models. However, the performance was not superior to RNNs, so we decided to stick with the latter. Last but not least, we tried fine-tuning techniques and trained our data on pre-trained models Devlin et al. (2018) but with no success. This type of encoding could not take apart the hypothesis from the knowledge base and the dense vectors the model produced were extremely similar to each other. As a result, there was no learning at all. We solved this problem by encoding the knowledge base and the hypothesis independently, but even then, the models were not able to outperform the other architectures.

A.4 Overall accuracy by types of inference

We present the detailed results from the experiment described in 4.1. Tables 10, 11, 12, and 13 show the overall performance results by types of inference for MLPs, RNNs, CNNs, and TRAs, respectively. The NNM column is the mean percentage of the model's output when taking into account all correct predictions, i.e., correct inferences that are not necessarily minimal. Moreover, in the last column, we present the average *Hamming distance* (HD) between the correct NNM predictions and the labels (the correct answer), *i.e.*, the average number of premises that are not needed. Note that for all architectures, this value is smaller than 2, which means that models (on average) do not select too many unneeded premises whenever they get the needed ones.

A.5 Generalization gap

We test on the training data for all architectures to check the generalization gap, i.e., the difference in performance on training versus test data. It can be seen from Table 14 that in this sense the models generalize very well (compare it with Table 3).

A.6 Permutation test

For this test, we train a model on a knowledge base $\mathcal{KB}_1$, and test it on a new knowledge base $\mathcal{KB}_2$. However, we count an output as correct if

Model	Layers
MLP	1 *Dense* layer with 2500 *units* and *tanh* activation
RNN	2 *SimpleRNN* layers with 250 *units* and *tanh* activation
CNN	1 *Conv1D* layer with 512 *filters*, a *kernel* of size 5, and *relu* activation 1 *MaxPooling1D* layer with a *pool* size of 3
TRA	1 *Embedding* layer (to learn the positions of constants and quantifiers) 1 *Encoder self-attention* layer: 1 *MultiHeadAttention* layer with 2 *heads* 1 Feed-forward network (3 hidden *Dense* layers with 32 *units* and *relu* activation) 1 *Dense* layer with 250 *units* and *tanh* activation

Table 9: Layers used in all architectures

Inf.	Best	Mean	SD	NNM	HD
1	80.6	46.8	17.9	55.5	1.2
2	92.0	80.9	13.2	83.3	1.1
3	99.2	89.6	8.1	92.9	1.0
4	93.0	78.9	15.9	88.7	1.1
5	100.0	95.0	8.3	96.8	1.0
6	100.0	89.5	11.2	90.0	1.0
7	99.4	88.1	15.4	93.4	1.0
Val.	93.9	83.2	13.1	88.9	1.1
Inv.	97.1	94.2	2.5	–	–
All	96.6	93.5	3.1	–	–

Table 10: Overall accuracy for MLP

Inf.	Best	Mean	SD	NNM	HD
1	78.4	58.3	11.3	64.3	1.7
2	96.6	92.5	2.5	93.1	1.2
3	96.4	92.3	3.4	92.5	1.0
4	93.0	90.6	1.4	96.2	1.1
5	100.0	97.9	2.9	98.4	1.0
6	100.0	93.3	5.8	93.3	0.0
7	100.0	98.3	1.3	99.0	1.0
Val.	94.3	92.0	1.3	94.4	1.2
Inv.	97.3	96.7	0.3	–	–
All	96.9	96.4	0.2	–	–

Table 12: Overall accuracy for CNN

Inf.	Best	Mean	SD	NNM	HD
1	77.8	58.6	9.7	60.7	1.0
2	96.5	92.1	3.6	92.8	1.2
3	99.3	96.6	1.8	96.9	1.0
4	96.1	92.0	2.2	96.1	1.1
5	100.0	99.5	1.2	99.5	0.0
6	100.0	97.7	2.0	97.7	0.0
7	99.4	97.9	1.1	98.9	1.0
Val.	95.9	93.5	1.3	95.3	1.1
Inv.	98.3	97.7	0.5	–	–
All	98.0	97.4	0.4	–	–

Table 11: Overall accuracy for RNN

Inf.	Best	Mean	SD	NNM	HD
1	74.1	62.8	8.4	72.1	1.4
2	96.5	90.8	3.6	92.9	1.0
3	96.4	92.6	2.8	94.8	1.3
4	99.6	96.1	3.7	97.9	1.0
5	100.0	94.3	5.2	97.0	1.0
6	100.0	98.6	2.3	99.3	1.0
7	98.8	96.0	3.1	97.5	1.0
Val.	96.6	93.6	2.9	95.7	1.1
Inv.	97.8	96.3	1.3	–	–
All	97.7	96.1	1.3	–	–

Table 13: Overall accuracy for TRA

Model	Inf.	Best	Mean	SD
MLP	Val.	98.9	90.5	10.1
	Inv.	99.6	98.4	1.3
	All	99.3	95.2	4.6
RNN	Val.	99.5	99.0	0.3
	Inv.	99.9	99.7	0.1
	All	99.6	99.4	0.1
CNN	Val.	99.4	98.8	0.3
	Inv.	99.8	99.7	0.1
	All	99.6	99.3	0.1
TRA	Val.	99.4	97.8	2.1
	Inv.	99.9	99.3	0.8
	All	99.6	98.7	1.2

Table 14: Test on the same data used for training

Model	Inf.	Best	Mean	SD
MLP	Val.	66.2	61.0	3.7
	Inv.	77.4	72.7	2.7
	All	75.1	70.6	2.9
RNN	Val.	14.3	4.3	3.1
	Inv.	30.6	11.4	6.7
	All	27.6	10.1	6.0
CNN	Val.	73.2	68.4	8.2
	Inv.	84.3	81.0	5.0
	All	82.2	78.7	5.6
TRA	Val.	98.3	97.4	0.6
	Inv.	98.0	97.4	0.5
	All	98.0	97.4	0.4

Table 15: Permutation test on new knowledge bases

it is correct for $\mathcal{KB}_1$. High performance on this test indicates that the model memorized the training base $\mathcal{KB}_1$, and ignores the part of the input corresponding to $\mathcal{KB}_2$.

We selected 3 knowledge bases and performed 6 tests, *i.e.*, trained models were tested on the other 2 knowledge bases. Table 15 shows the accuracies calculated in the way described above. TRAs memorize the training base almost perfectly, while RNNs do not memorize in this way.

A.7 Principle of Contradiction, non-emptiness of denotations, symmetry

For these tests, we search the output for the following pairs of hypotheses: (1) $\{H, \overline{H}\}$; (2) $\{Aab, Iab\}$; (3) $\{Iab, Iba\}$ and $\{Eab, Eba\}$. Then we calculate the percentage of pairs that confirm (1) Principle of Contradiction (2) Non-emptyness of denotations (i.e., Aab implies Iab), and (3) symmetry of formulas Iab, Eab. The results are shown in Table 16 for tests on a new knowledge base, and Table 17 for tests on the same knowledge base (*i.e.*, its own test dataset). Apparently, RNNs poorly generalize Principle of Contradiction on a new knowledge base.

Model	Pair	Highest	Mean	SD
MLP	(1)	94.8	92.9	1.1
	(2)	100.0	100.0	0.1
	(3)	93.6	92.3	0.7
RNN	(1)	75.7	59.8	5.6
	(2)	100.0	100.0	0.0
	(3)	98.3	94.2	1.9
CNN	(1)	100.0	99.3	1.5
	(2)	100.0	99.7	0.4
	(3)	100.0	98.5	1.4
TRA	(1)	99.9	99.7	0.2
	(2)	100.0	99.8	0.2
	(3)	99.8	99.6	0.2

Table 16: Pairs on new KBs. (1) valid/invalid $\{H, \overline{H}\}$, (2) valid/valid $\{Aab, Iab\}$, (3) valid/valid $\{Iab, Iba\}$, $\{Eab, Eba\}$

Model	Pair	Highest	Mean	SD
MLP	(1)	99.8	99.4	0.3
	(2)	100.0	99.9	0.1
	(3)	98.5	97.6	0.8
RNN	(1)	99.9	99.9	0.0
	(2)	100.0	100.0	0.0
	(3)	99.7	99.2	0.2
CNN	(1)	99.7	99.7	0.1
	(2)	100.0	100.0	0.0
	(3)	99.1	98.9	0.2
TRA	(1)	100.0	99.8	0.2
	(2)	100.0	99.7	0.1
	(3)	99.8	99.4	0.5

Table 17: Pairs on test data. (1) valid/invalid $\{H, \overline{H}\}$, (2) valid/valid $\{Aab, Iab\}$, (3) valid/valid $\{Iab, Iba\}$, $\{Eab, Eba\}$

$\mathcal{F} \cup \{\overline{F}\}$	$\overline{F}$	$\mathcal{F} \vdash F$
$\{Aa-b, Oab\}$	Aab	$\{Oab\} \vdash Oab$
	Aax_1	$\{Ax_1 - b, Oab\} \vdash Oax_1$
	$Ax_ix_{i+1}{}^{(1)}$	$\{Aa - x_i, Ax_{i+1} - b, Oab\} \vdash Ox_ix_{i+1}$
	Ax_mb	$\{Aa - x_m, Oab\} \vdash Ox_mb$
	Oab	$\{Aa - b\} \vdash Aab$
$\{Aa-b, Aa-c, Ebc\}$	Aab	$\{Aa - c, Ebc\} \vdash Oab$
	Aac	$\{Aa - b, Ebc\} \vdash Oac$
	Aax_1	$\{Ax_1 - b, Aa - c, Ebc\} \vdash Oax_1$
	Aay_1	$\{Aa - b, Ay_1 - c, Ebc\} \vdash Oay_1$
	$Ax_ix_{i+1}{}^{(1)}$	$\{Aa - x_i, Ax_{i+1} - b, Aa - c, Ebc\} \vdash Ox_ix_{i+1}$
	$Ay_iy_{i+1}{}^{(2)}$	$\{Aa - b, Aa - y_i, Ay_{i+1} - c, Ebc\} \vdash Oy_iy_{i+1}$
	Ax_mb	$\{Aa - x_m, Aa - c, Ebc\} \vdash Ox_mb$
	Ay_nc	$\{Aa - b, Aa - y_n, Ebc\} \vdash Oy_nc$
	Ebc	$\{Aa - b, Aa - c\} \vdash Ibc$
$\{Aa-b, Ac-d, Iac, Ebd\}$	Aab	$\{Ac - d, Iac, Ebd\} \vdash Oab$
	Acd	$\{Aa - b, Iac, Ebd\} \vdash Ocd$
	Aax_1	$\{Ax_1 - b, Ac - d, Iac, Ebd\} \vdash Oax_1$
	Acy_1	$\{Aa - b, Ay_1 - d, Iac, Ebd\} \vdash Ocy_1$
	$Ax_ix_{i+1}{}^{(1)}$	$\{Aa - x_i, Ax_{i+1} - b, Ac - d, Iac, Ebd\} \vdash Ox_ix_{i+1}$
	$Ay_iy_{i+1}{}^{(2)}$	$\{Aa - b, Ac - y_i, Ay_{i+1} - d, Iac, Ebd\} \vdash Oy_iy_{i+1}$
	Ax_mb	$\{Aa - x_m, Ac - d, Iac, Ebd\} \vdash Ox_mb$
	Ay_nd	$\{Aa - b, Ac - y_n, Iac, Ebd\} \vdash Oy_nd$
	Iac	$\{Aa - b, Ac - d, Ebd\} \vdash Eac$
	Ebd	$\{Aa - b, Ac - d, Iac\} \vdash Ibd$

$^{(1)}\forall i.1 \leq i < m$ $^{(2)}\forall i.1 \leq i < n$

Table 18: Construction of inferences