# A HINT FROM ARITHMETIC: ON SYSTEMATIC GENERALIZATION OF PERCEPTION, SYNTAX, AND SEMANTICS

**Qing Li, Siyuan Huang, Yining Hong, Yixin Zhu, Ying Nian Wu, Song-Chun Zhu**
University of California, Los Angeles
{liqing, huangsiyuan, yininghong, yixin.zhu}@ucla.edu, {ywu,sczhu}@stat.ucla.edu

## ABSTRACT

Inspired by humans' remarkable ability to master arithmetic and generalize to unseen problems, we present a new dataset, HINT, to study machines' capability of learning generalizable concepts at three different levels: *perception*, *syntax*, and *semantics*. In particular, concepts in HINT, including both digits and operators, are required to learn in a weakly-supervised fashion: Only the final results of handwriting expressions are provided as supervision. Learning agents need to reckon how concepts are perceived from raw signals such as images (*i.e.*, perception), how multiple concepts are structurally combined to form a valid expression (*i.e.*, syntax), and how concepts are realized to afford various reasoning tasks (*i.e.*, semantics). With a focus on systematic generalization, we carefully design a five-fold test set to evaluate both the *interpolation* and the *extrapolation* of learned concepts. To tackle this challenging problem, we propose a neural-symbolic system by integrating neural networks with grammar parsing and program synthesis, learned by a novel deduction–abduction strategy. In the experiments, the proposed neural-symbolic system demonstrates strong generalization capability and significantly outperforms end-to-end neural methods like RNN and Transformer. An additional preliminary few-shot study also indicates that the proposed neural-symbolic system can quickly learn new concepts with limited examples.[1]

## 1 INTRODUCTION

Humans possess a versatile mechanism for learning concepts (Firestone & Scholl, 2016). Take the arithmetic examples in Fig. 1: When we master concepts like digits and operators, we not only know how to recognize, write, and pronounce them—what these concepts mean at the *perceptual* level, but also know how to compose them into valid expressions—at the *syntactic* level, and how to calculate the results by reasoning over these concepts—at the *semantic* level. Learning concepts rely heavily on these three-level interweaving meanings. Such observation also conforms with the classic view of human cognition, which postulates at least three distinct levels of organizations in computation systems (Pylyshyn, 1984). Crucially, a unique property of human concept learning is its systematic generalization (Xie et al., 2021; Lake et al., 2017; Fodor et al., 1988). Once we master arithmetic using short expressions with small numbers, we can generalize to novel, long expressions with unseen handwriting and large numbers.

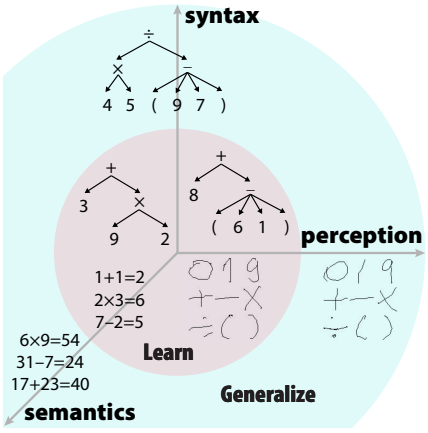



Figure 1: **Concept learning and generalization on perception, syntax, and semantics.**

To examine the versatile humanlike capabilities of concept learning with a focus on systematic generalization, we introduce a new benchmark HINT, Handwritten arithmetic with INTegers. The task of HINT is intuitive: Machines take as input images of handwritten expressions and predict the final results of expressions, restricted in the integer space. The task of HINT is also challenging: Concepts

[1]Check the project website: tinyurl.com/iclr21hint for the dataset, the code, and a demo video.

in HINT, including digits and operators, are learned in a weakly-supervised manner. Using final results as the only supervision, the three-level meanings are presumably intertwined during learning. To provide a holistic and rigorous test on whether learning machines can generalize the learned concepts, we carefully design an evaluation scheme to test generalization capabilities (*i.e.*, interpolation and extrapolation) at different levels of meanings (*i.e.*, perception, syntax, and semantics).

Inspired by the superb generalization capability demonstrated in symbolic systems with combinatorial structure (Fodor et al., 1988) and recent advances in neural-symbolic integration (Li et al., 2020a; Yi et al., 2018; Manhaeve et al., 2018), we propose an Arithmetic Neural-Symbolic (ANS) system to approach the HINT challenge. The proposed ANS system integrates the learning of perception, syntax, and semantics in a principled framework; see an illustration in Fig. 2. Specifically, we first utilize ResNet-18 (He et al., 2016) as a perception module to translate a handwritten expression into a symbolic sequence. This symbolic sequence is then parsed by a transition-based neural dependency parser (Chen & Manning, 2014), which encodes the syntax of concepts. Finally, we adopt functional programs to realize the semantic meaning of concepts, thus view learning semantics as program induction (Ellis et al., 2020). We derive a novel *deduction-abduction* strategy to coordinate the learning of different modules. During learning, the system first performs greedy deduction over these modules to propose an initial, rough solution, which is likely to produce a wrong result. A one-step abduction over perception, syntax, and semantics is then applied in a top-down manner to rectify the initial solution. The revised solution provides *pseudo* supervisions on the intermediate values and representations, which are then used to train each module individually.

Evaluated on HINT, ANS exhibits strong systematic generalization with an overall accuracy of 72%, outperforming end-to-end neural methods by nearly 33%. Results also indicate the strong generalization of ANS relies on its underlying *symbol system* (Fodor et al., 1988) encoded with *recursive* priors, which facilitate the extrapolation on syntax and semantics. A preliminary study of few-shot learning further demonstrates that ANS can quickly learn new concepts with limited examples, obtaining an accuracy of 62% on four new concepts with a hundred training examples.

## 2 THE HINT BENCHMARK

**Task Definition** The task of HINT is intuitive and straightforward: It is tasked to predict the final results of handwritten arithmetic expressions in a weakly-supervised manner. Only the final results are given as supervision; all intermediate values and representations are latent, including symbolic expressions, parse trees, and execution traces.

**Data Generation** The data generation process follows three steps; see Fig. S1 for an illustration. First, we extract handwritten images from `CROHME` to obtain primitive concepts $\{0 \sim 9, +, -, \times, \div, (, )\}$. Second, we randomly sample *prefix* expressions and convert them to *infix* expressions with necessary parentheses based on the operator precedence; we only allow single-digit numbers in expressions. These symbolic expressions are fed into a solver to calculate the final results. Third, we randomly sample handwritten images for symbols in an expression and concatenate them to construct final handwritten expressions. We only keep the handwritten expressions as input and the corresponding final results as supervision; all intermediate results are discarded.

**Train and Evaluation** To evaluate how well the learned concepts are systematically generalized, we replace the typical *i.i.d.* train/test split with a carefully designed evaluation scheme:

$$D_{train} \subset \mathcal{D}_{train} = \{(x, y) : |x| \leqslant 10, \max(v) \leqslant 100\}, D_{test} = D_{test}^{(1)} \cup D_{test}^{(2)} \cup D_{test}^{(3)} \cup D_{test}^{(4)} \cup D_{test}^{(5)},$$

| | |
|---|---|
| $D_{test}^{(1)} = D_{train},$ | no generalization on either syntax or semantics |
| $D_{test}^{(2)} \subset \mathcal{D}_{train} \backslash D_{train},$ | interpolation on both syntax and semantics |
| $D_{test}^{(3)} \subset \{(x, y) : \lvert x\rvert \leqslant 10, \max(v) > 100\},$ | interpolation on syntax and extrapolation on semantics |
| $D_{test}^{(4)} \subset \{(x, y) : \lvert x\rvert > 10, \max(v) \leqslant 100\},$ | extrapolation on syntax and interpolation on semantics |
| $D_{test}^{(5)} \subset \{(x, y) : \lvert x\rvert > 10, \max(v) > 100\},$ | extrapolation on both syntax and semantics |

where $x$ is the handwritten expression, $|x|$ its number of operators, $y$ the final result, and $v$ all the intermediate values generated when calculating the final result. All subsets in the test set requires generalization on perception, since all images in the test set are unseen in training.

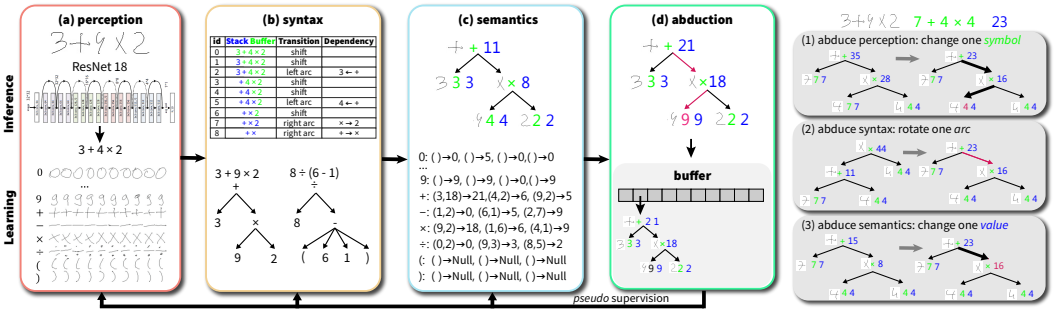



Figure 2: **The Arithmetic Neural-Symbolic model (ANS).** . During inference, the model performs greedy deduction over **(a) perception , (b) syntax, (c) semantics**, and directly proposes a solution. During learning, the proposed solution is further revised by performing **(d) abduction** based on the ground-truth supervision. The updated solution is stored in a buffer, providing *pseudo* supervisions to train three modules individually. Each node in the solution tree is a triplet of (image, symbol, value). Parts revised in abduction are highlighted in red.

## 3 Arithmetic Neural-Symbolic (ANS) Model

To approach the HINT challenge, we propose a neural-symbolic model ANS, which integrates the learning of perception, syntax, and semantics in a principled framework; see an illustration in Fig. 2.

The **perception** module is a standard ResNet-18 (He et al., 2016) to map a handwritten expression $x$ into a symbolic expression $s$. Since disentangling visual symbols from handwritten expressions is trivial in this domain, we assume the input as a sequence of handwritten images, where each image contains one symbol. Since learning it from scratch is prohibitively challenging, the ResNet-18 is pre-trained unsupervisedly (Van Gansbeke et al., 2020) on unlabeled handwritten images.

To parse the symbolic sequence into a **syntactic** tree, we adopt a greedy transition-based neural dependency parser (Chen & Manning, 2014), commonly used for parsing natural language sentences. The transition-based dependency parser relies on a state machine that defines the possible transitions to parse the input sequence into a dependency tree; see panel (b) of Fig. 2. The learning process induces a model to predict the next transition in the state machine based on the transition history. The parsing process constructs the optimal sequence of transitions for the input sequence. A dependency parser for arithmetic expressions is essentially approximating the Shunting-yard algorithm.

To learn **semantics** as programs, we start from DreamCoder (Ellis et al., 2020), which embodies a wake-sleep Bayesian program induction approach to progressively learn multiple tasks from a set of domain primitives and input-out pairs for each task. For arithmetic reasoning, the Peano axioms (Peano, 1889) define four primitives: (1) `0`; (2) `inc`: $a \to a+1$; (3) `dec`: $a \to \max(0, a-1)$; (4) `if`: $(a, b, c) \to b$ (if $a$ is 0) or $c$ (else). This set of primitives is augmented with a recursion primitive, `Y`-combinator (*a.k.a.*, fixed-point combinator). The `Y`-combinator enables the derivation of recursive functions and is the crux of extrapolating to large numbers.

The **abduction** is applied over perception, syntax, and semantics in a top-down manner to rectify the initial solution, as illustrated in Fig. 2. The revised solution provides *pseudo* supervision on the intermediate values and representations, which are then used to train each module individually.

Please refer to Appendix B for more details on the model and experimental settings.

## 4 Results and Discussions

### 4.1 NeSy v.s. E2E NeuralNets

We compare the performance of the proposed neural-symbolic model ANS with end-to-end neural baselines on HINT. As shown in Table 1, both BiGRU and TRAN obtain high accuracy on the test subset 1, which indicates that they can generalize over perception very well. However, their performances drop significantly on the test subsets 2∼5, which require systematic generalization over syntax and semantics. Notably, their accuracy is less than 10% on test subsets 3 and 5 that involve larger numbers compared to the training set. This result indicates that the pure neural models do not learn the semantics of concepts in a generalizable way and fail to extrapolate to large numbers. In contrast, the proposed ANS model consistently outperforms BiGRU and TRAN by at least 30 absolute percent across all test subsets

Table 1: The performance comparison of ANS and end-to-end neural networks, *i.e.*, GRU (BiGRU) and Transformer (TRAN).

| Input | Model | Test Accuracy (%) | | | | | |
|---|---|---|---|---|---|---|---|
| | | Overall | 1 | 2 | 3 | 4 | 5 |
| Symbol (Embedding) | BiGRU | 49.71 | 97.05 | 63.67 | 11.58 | 52.41 | 12.57 |
| | TRAN | 34.58 | 98.31 | 29.79 | 2.91 | 26.39 | 2.76 |
| | ANS | **88.36** | **99.26** | **97.56** | **84.66** | **87.65** | **65.37** |
| Image (ResNet-18) | BiGRU | 39.39 | 87.02 | 46.17 | 6.51 | 40.44 | 6.47 |
| | TRAN | 32.95 | 87.31 | 30.74 | 2.67 | 31.17 | 2.55 |
| | ANS | **71.97** | **89.10** | **84.29** | **66.77** | **68.19** | **40.73** |

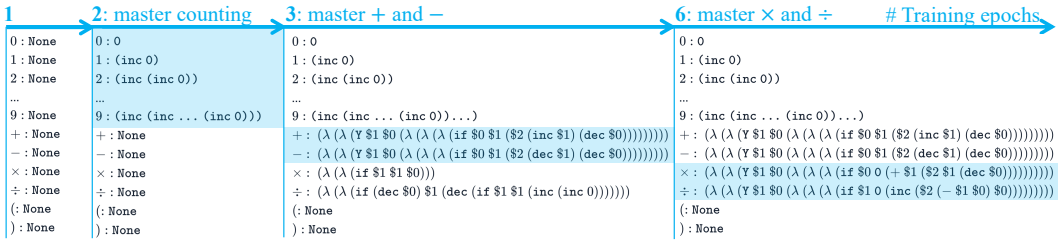



Figure 3: **The evolution of semantics in ANS from initial primitives {`0`, `inc`, `dec`, `if`, `Y`}.**

2∼5. This superb performance demonstrates the strong systematic generalization of ANS, including both interpolation and extrapolation w.r.t. syntax and semantics.

**How do models extrapolate?** Among the generalization capability, we are particularly interested in extrapolation. Based on the experimental results, we firmly believe that the key is *recursion*. In ANS, the extrapolation on syntax is achieved by the transition system of the dependency parser, which recursively applies transition actions to parse arbitrarily long expressions. The extrapolation on semantics is realized by the recursion primitive, *i.e.*, `Y`-combinator. It allows programs to represent recursive functions, which can decompose large numbers into smaller ones by recursively invoking themselves. For BiGRU, although the recurrent structure in its hidden cells serves as a recursive prior on syntax, no such prior in its representation for semantics. This deficiency explains why BiGRU would achieve a decent accuracy (40.44%) on the test subset 3 (extrapolation only on syntax) but a much lower accuracy (6.51%) on the test subset 4 (extrapolation only on semantics). Taken together, these observations strongly imply that the recursive prior on task-specific representations is the crux of extrapolation, which is also in line with the recent analysis of Graph Neural Network, where it successfully extrapolates algorithmic tasks due to the *task-specific non-linearities* in the architecture or features (Xu et al., 2020b;a).

### 4.2 Ablation Study

Table 2: **Ablation study on ANS.** ✓ indicates that the ground-truth labels are given during training.

| Training Setting | | | Test Accuracy (%) | | | | | |
|---|---|---|---|---|---|---|---|---|
| Per. | Syn. | Sem. | Overall | 1 | 2 | 3 | 4 | 5 |
| | | ✓ | 71.97 | 89.10 | 84.29 | 66.77 | 68.19 | 40.73 |
| | ✓ | | 86.44 | 94.53 | 91.62 | 89.58 | 78.22 | 71.18 |
| ✓ | | | 80.14 | 92.51 | 90.16 | 71.32 | 84.27 | 56.27 |
| ✓ | ✓ | | 88.36 | 99.26 | 97.56 | 84.66 | 87.65 | 65.37 |
| ✓ | | ✓ | 97.81 | 100.00 | 100.00 | 96.66 | 100.00 | 90.97 |
| | ✓ | ✓ | 95.84 | 99.60 | 98.23 | 98.09 | 91.50 | 88.20 |
| | | | 88.93 | 94.30 | 92.19 | 90.06 | 82.99 | 80.88 |

Table 2 shows an ablation study on the proposed ANS model. In general, providing the ground-truth meaning of concepts can ease the learning and lead to higher test accuracy. Among the three levels of concepts, perception is the hardest to learn since the handwriting images possess a large variance in terms of the visual appearance. The syntax and semantics are relatively easier to learn, since the recursive prior of the transition-based dependency parser and `Y`-combinator fits the task well.

Fig. 3 illustrates the typical pattern of the evolution of semantics in ANS. This pattern is highly in accord with how children learn arithmetic in developmental psychology (Carpenter et al., 1999): The model first masters the semantics of digits as `counting`, then learns $+$ and $-$ as recursive counting, and finally it figures out how to define $\times$ and $\div$ based on the learned programs for $+$ and $-$. Crucially, $\times$ and $\div$ are impossible to be correctly learned before mastering $+$ and $-$. The model is endowed with such an incremental learning capability since the program induction module allows the semantics of concepts to be built compositionally from those learned earlier (Ellis et al., 2020).

### 4.3 Few-shot Concept Learning

Table 3: **Few-shot concept learning with ANS.**

| Per. | Syn. | Sem. | Test Accuracy (%) | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | Overall | 1 | 2 | 3 | 4 | 5 |
| $\alpha$ ⩔ | 1 | $\max(x,y)$ | 64.08 | 70.91 | 81.98 | 70.79 | 50.56 | 40.66 |
| $\beta$ ⩓ | 1 | $\min(x,y)$ | 72.45 | 85.45 | 83.93 | 81.82 | 65.91 | 40.22 |
| $\gamma$ ⩒ | 2 | $(x+y)/2$ | 56.73 | 76.36 | 70.09 | 61.80 | 41.94 | 27.47 |
| $\phi$ ⩔ | 2 | $xy-(x+y)$ | 54.40 | 76.36 | 68.81 | 41.35 | 56.04 | 22.09 |
| avg. | - | - | 61.92 | 77.27 | 76.20 | 63.94 | 53.61 | 32.61 |

We further conduct a preliminary study of few-shot learning to demonstrate the ANS's potential in learning new concepts with limited examples. As shown in Table 3, we define four new concepts with common semantics. Their visual appearances are denoted by four unseen handwritten symbols $\{\alpha, \beta, \gamma, \phi\}$, and their syntax is decided by their precedence (*i.e.*, 1 is for $\{+, -\}$ and 2 is for $\{\times, \div\}$). We randomly sample a hundred examples from short to long expressions for training each new concept and fine-tune the ANS model on the new training data. Table 3 shows the test accuracy for each new concept. The proposed ANS model obtains a decent performance with an average overall accuracy of 61.92%. Concepts with more complex semantics ($\{\gamma, \phi\}$) are generally harder to learn than those with simpler semantics ($\{\alpha, \beta\}$).

**Acknowledgements.** The authors thank Sirui Xie and Chi Zhang from UCLA CS Department for helpful discussions. The work reported herein was supported by ONR N00014-19-1-2153, ONR MURI N00014-16-1-2007, and DARPA XAI N66001-17-2-4029.

## REFERENCES

Sebastian Bader, Artur S d'Avila Garcez, and P Hitzler. Extracting propositional rules from feed-forward neural networks by means of binary decision diagrams. In *5th International Workshop on Neural-Symbolic Learning and Reasoning, NeSy*, 2009.

Dzmitry Bahdanau, Shikhar Murty, Michael Noukhovitch, Thien Huu Nguyen, Harm de Vries, and Aaron Courville. Systematic generalization: what is required and can it be learned? In *International Conference on Learning Representations (ICLR)*, 2018.

Matej Balog, Alexander L Gaunt, Marc Brockschmidt, Sebastian Nowozin, and Daniel Tarlow. Deepcoder: Learning to write programs. In *International Conference on Learning Representations (ICLR)*, 2017.

Eli Bingham, Jonathan P Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D Goodman. Pyro: Deep universal probabilistic programming. *Journal of Machine Learning Research*, 20(1):973–978, 2019.

Christopher P Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. Monet: Unsupervised scene decomposition and representation. In *International Conference on Learning Representations (ICLR)*, 2019.

Bob Carpenter, Andrew Gelman, Matthew D Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus A Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: a probabilistic programming language. *Grantee Submission*, 76(1):1–32, 2017.

Thomas P Carpenter, Elizabeth Fennema, M Loef Franke, Linda Levi, and Susan B Empson. Children's mathematics. *Cognitively Guided*, 1999.

Danqi Chen and Christopher D Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

Yixin Chen, Siyuan Huang, Tao Yuan, Siyuan Qi, Yixin Zhu, and Song-Chun Zhu. Holistic++ scene understanding: Single-view 3d holistic scene parsing and human pose estimation with human-object interaction and physical commonsense. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2019.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Wang-Zhou Dai, Qiuling Xu, Yang Yu, and Zhi-Hua Zhou. Bridging machine learning and logical reasoning by abductive learning. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Jacob Devlin, Jonathan Uesato, Surya Bhupatiraju, Rishabh Singh, Abdel-rahman Mohamed, and Pushmeet Kohli. Robustfill: Neural program learning under noisy i/o. In *Proceedings of International Conference on Machine Learning (ICML)*, 2017.

Mark Edmonds, Feng Gao, Hangxin Liu, Xu Xie, Siyuan Qi, Brandon Rothrock, Yixin Zhu, Ying Nian Wu, Hongjing Lu, and Song-Chun Zhu. A tale of two explanations: Enhancing human trust by explaining robot behavior. *Science Robotics*, 4(37), 2019.

Kevin Ellis, Daniel Ritchie, Armando Solar-Lezama, and Joshua B Tenenbaum. Learning to infer graphics programs from hand-drawn images. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2018a.

Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sable-Meyer, Luc Cary, Lucas Morales, Luke Hewitt, Armando Solar-Lezama, and Joshua B Tenenbaum. Dreamcoder: Growing generalizable, interpretable knowledge with wake-sleep bayesian program learning. *arXiv preprint arXiv:2006.08381*, 2020.

Kevin M Ellis, Lucas E Morales, Mathias Sablé-Meyer, Armando Solar Lezama, and Joshua B Tenenbaum. Library learning for neurally-guided bayesian program induction. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2018b.

Richard Evans and Edward Grefenstette. Learning explanatory rules from noisy data. *Journal of Artificial Intelligence Research*, 61:1–64, 2018.

Chaz Firestone and Brian J Scholl. Cognition does not affect perception: Evaluating the evidence for "top-down" effects. *Behavioral and Brain Sciences*, 39, 2016.

Jerry A Fodor. *The language of thought*, volume 5. Harvard university press, 1975.

Jerry A Fodor, Zenon W Pylyshyn, et al. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71, 1988.

Artur SD'Avila Garcez, Luis C Lamb, and Dov M Gabbay. *Neural-symbolic cognitive reasoning*. Springer Science & Business Media, 2008.

Hong Ge, Kai Xu, and Zoubin Ghahramani. Turing: A language for flexible probabilistic inference. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018.

Zoubin Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553): 452–459, 2015.

Jonathan Gordon, David Lopez-Paz, Marco Baroni, and Diane Bouchacourt. Permutation equivariant models for compositional generalization in language. In *International Conference on Learning Representations (ICLR)*, 2019.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

Ulf Grenander. *General pattern theory-A mathematical study of regular structures*. Clarendon Press, 1993.

Abhinav Gupta, Praveen Srinivasan, Jianbo Shi, and Larry S Davis. Understanding videos, constructing plots learning a visually grounded storyline model from annotated videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition. *Signal Processing Magazine*, 29(6):82–97, 2012.

Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

Steven Holtzen, Guy Van den Broeck, and Todd Millstein. Scaling exact inference for discrete probabilistic programs. *Proceedings of the ACM on Programming Languages*, 4(OOPSLA):1–31, 2020.

Siyuan Huang, Siyuan Qi, Yinxue Xiao, Yixin Zhu, Ying Nian Wu, and Song-Chun Zhu. Cooperative holistic scene understanding: Unifying 3d object, layout, and camera pose estimation. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2018a.

Siyuan Huang, Siyuan Qi, Yixin Zhu, Yinxue Xiao, Yuanlu Xu, and Song-Chun Zhu. Holistic 3d scene parsing and reconstruction from a single rgb image. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2018b.

Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

baoxiong Jia, Yixin Chen, Siyuan Huang, Yixin Zhu, and Song-Chun Zhu. Lemma: A multi-view dataset for learning multi-agent multi-task activities. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2020.

Chenfanfu Jiang, Siyuan Qi, Yixin Zhu, Siyuan Huang, Jenny Lin, Lap-Fai Yu, Demetri Terzopoulos, and Song-Chun Zhu. Configurable 3d scene synthesis and 2d image rendering with per-pixel ground truth using stochastic grammars. *International Journal of Computer Vision (IJCV)*, 126 (9):920–941, 2018.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, et al. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations (ICLR)*, 2019.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Nikita Kitaev and Dan Klein. Constituency parsing with a self-attentive encoder. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2018.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2012.

Tejas D Kulkarni, Pushmeet Kohli, Joshua B Tenenbaum, and Vikash Mansinghka. Picture: A probabilistic programming language for scene perception. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Brenden M. Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of International Conference on Machine Learning (ICML)*, 2018.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40, 2017.

Guillaume Lample and François Charton. Deep learning for symbolic mathematics. In *International Conference on Learning Representations (ICLR)*, 2020.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

Dennis Lee, Christian Szegedy, Markus N Rabe, Sarah M Loos, and Kshitij Bansal. Mathematical reasoning in latent space. In *International Conference on Learning Representations (ICLR)*, 2020.

Qing Li, Siyuan Huang, Yining Hong, Yixin Chen, Ying Nian Wu, and Song-Chun Zhu. Closed loop neural-symbolic learning via integrating neural perception, grammar parsing, and symbolic reasoning. In *Proceedings of International Conference on Machine Learning (ICML)*, 2020a.

Qing Li, Siyuan Huang, Yining Hong, and Song-Chun Zhu. A competence-aware curriculum for visual concepts learning via question answering. *Proceedings of European Conference on Computer Vision (ECCV)*, 2020b.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. *arXiv preprint arXiv:1611.00020*, 2016.

Hangxin Liu, Yaofang Zhang, Wenwen Si, Xu Xie, Yixin Zhu, and Song-Chun Zhu. Interactive robot knowledge patching using augmented reality. In *Proceedings of International Conference on Robotics and Automation (ICRA)*, 2018.

Hangxin Liu, Chi Zhang, Yixin Zhu, Chenfanfu Jiang, and Song-Chun Zhu. Mirroring without overimitation: Learning functionally equivalent manipulation actions. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

John W Lloyd. *Foundations of logic programming*. Springer Science & Business Media, 2012.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *arXiv preprint arXiv:2006.15055*, 2020.

Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Deepproblog: Neural probabilistic logic programming. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In *International Conference on Learning Representations (ICLR)*, 2018.

Stephen Muggleton and Luc De Raedt. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19:629–679, 1994.

Emilio Parisotto, Abdel-rahman Mohamed, Rishabh Singh, Lihong Li, Dengyong Zhou, and Pushmeet Kohli. Neuro-symbolic program synthesis. In *International Conference on Learning Representations (ICLR)*, 2016.

Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. Specaugment: A simple data augmentation method for automatic speech recognition. In *Interspeech*, 2019.

Giuseppe Peano. *Arithmetices principia: Nova methodo exposita*. Fratres Bocca, 1889.

Zenon W Pylyshyn. Computation and cognition: Towards a foundation for cognitive science, 1984.

Siyuan Qi, Baoxiong Jia, and Song-Chun Zhu. Generalized earley parser: Bridging symbolic grammars and sequence data for future prediction. In *Proceedings of International Conference on Machine Learning (ICML)*, 2018a.

Siyuan Qi, Yixin Zhu, Siyuan Huang, Chenfanfu Jiang, and Song-Chun Zhu. Human-centric indoor scene synthesis using stochastic grammar. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018b.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

Ron Sun. *Integrating rules and connectionism for robust commonsense reasoning*. John Wiley & Sons, Inc., 1994.

Zhuowen Tu, Xiangrong Chen, Alan L Yuille, and Song-Chun Zhu. Image parsing: Unifying segmentation, detection, and recognition. *International Journal of Computer Vision (IJCV)*, 63(2): 113–140, 2005.

Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Scan: Learning to classify images without labels. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Sirui Xie, Xiaojian Ma, Peiyu Yu, Yixin Zhu, Ying Nian Wu, and Song-Chun Zhu. Halma: Humanlike abstraction learning meets affordance in rapid problem solving. *arXiv preprint arXiv:2102.11344*, 2021.

Xu Xie, Hangxin Liu, Mark Edmonds, Feng Gao, Siyuan Qi, Yixin Zhu, Brandon Rothrock, and Song-Chun Zhu. Unsupervised learning of hierarchical models for hand-object interactions. In *Proceedings of International Conference on Robotics and Automation (ICRA)*, 2018.

Keyulu Xu, Jingling Li, Mozhi Zhang, Simon S Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. What can neural networks reason about? In *International Conference on Learning Representations (ICLR)*, 2020a.

Keyulu Xu, Jingling Li, Mozhi Zhang, Simon S Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. How neural networks extrapolate: From feedforward to graph neural networks. *arXiv preprint arXiv:2009.11848*, 2020b.

Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Josh Tenenbaum. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Pengcheng Yin, Chunting Zhou, Junxian He, and Graham Neubig. Structvae: Tree-structured latent variable models for semi-supervised semantic parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2018.

Tao Yuan, Hangxin Liu, Lifeng Fan, Zilong Zheng, Tao Gao, Yixin Zhu, and Song-Chun Zhu. Joint inference of states, robot knowledge, and human (false-)beliefs. In *Proceedings of International Conference on Robotics and Automation (ICRA)*, 2020.

Chi Zhang, Feng Gao, Baoxiong Jia, Yixin Zhu, and Song-Chun Zhu. Raven: A dataset for relational and analogical visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019a.

Chi Zhang, Baoxiong Jia, Feng Gao, Yixin Zhu, Hongjing Lu, and Song-Chun Zhu. Learning perceptual inference by contrasting. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2019b.

Wenhe Zhang, Chi Zhang, Yixin Zhu, and Song-Chun Zhu. Machine number sense: A dataset of visual arithmetic problems for abstract and relational reasoning. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, 2020a.

Zhenliang Zhang, Yixin Zhu, and Song-Chun Zhu. Graph-based hierarchical knowledge representation for robot task transfer from virtual to physical world. In *Proceedings of International Conference on Intelligent Robots and Systems (IROS)*, 2020b.

Yibiao Zhao and Song-Chun Zhu. Image parsing with stochastic scene grammar. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2011.

Zhi-Hua Zhou. Abductive learning: towards bridging machine learning and logical reasoning. *Science China Information Sciences*, 62:1–3, 2019.

Song-Chun Zhu, David Mumford, et al. A stochastic grammar of images. *Foundations and Trends® in Computer Graphics and Vision*, 2(4):259–362, 2007.

# A THE HINT DATASET

The data generation process follows the pipeline illustrated in Fig. S1. The syntax of the *infix* expressions can be fully described by the context-free grammar depicted in Table S1. When generating the HINT dataset, we ensure that (i) all handwritten images in the test set are unseen in training, (ii) at most 1,000 samples are generated for each number of operators in expressions. In total, the training and test set includes 11,170 and 48,910 samples, respectively. Subsets in the test set are balanced to be 23%, 23%, 22%, 16%, and 16%. Fig. S2 visualizes several randomly selected examples from the proposed HINT dataset.

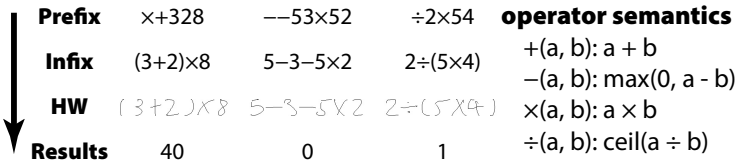



Figure S1: **The data generation pipeline.**

Table S1: **Context-free grammar for arithmetic expressions.**

| G = (V, Σ, R, S) |
|---|
| V= {S, Expression, Term, Factor, Number} |
| $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -, \times, \div, (, )\}$. |
| S is the start symbol. |
| R = {S → Expression |
| Expression → Term \| Expression + Term \| Expression - Term |
| Term → Factor \| Term × Factor \| Term ÷ Factor |
| Factor → ( Expression ) \| Number |
| Number → 0\|1\|2\|3...\|9 } |

# B A NEURAL-SYMBOLIC APPROACH

Below we first describe a general framework from a probabilistic perspective for learning the HINT task as a neural-symbolic approach. This general framework implies a *symbol system* with combinatorial syntactic and semantic structures, initially introduced by (Fodor et al., 1988), as a feasible representation of the human mind. Such a symbol system provides a principled integration of perception, syntax, and semantics. Guided by this general framework, we next provide a concrete instantiation of such a neural-symbolic system and introduce a novel deduction-abduction strategy to learn it with weak supervision; see Fig. 2 for overview.

## B.1 A GENERAL FRAMEWORK

Given a neural-symbolic system, let $x \in \Omega_x$ denote the input (images of handwritten expression in the HINT dataset), $s \in \Omega_s$ the symbolic expression, $pt \in \Omega_t$ the parse tree of the symbolic expression, $et \in \Omega_e$ the execution trace, and $y \in \Omega_y$ the output. During learning, $(x, y)$ are observed but $(s, pt, et)$ are latent. The likelihood of the observation $(x, y)$ marginalized over $(s, pt, et)$ can be decomposed as:

$$\begin{aligned} p(y|x;\Theta) &= \sum_{s,pt,et} p(s, pt, et, y|x;\Theta) \\ &= \sum_{s,pt,et} p(s|x;\theta_p)p(pt|s;\theta_s)p(et|pt;\theta_l)p(y|et), \end{aligned} \tag{1}$$

where (i) $s|x$ denotes the process of perceiving symbols from raw signals, guided by the perceptual model $\theta_p$ of learned concepts; (ii) $pt|s$ denotes the process of parsing the symbolic expression into a parse tree, guided by the syntactic model $\theta_s$; (iii) $et|pt$ denotes the process of reasoning over the parse tree, guided by the semantic model $\theta_l$; and (iv) $y|et$ is a deterministic process: If the final output of $et$ equals to $y$, $p(y|et) = 1$, otherwise 0.

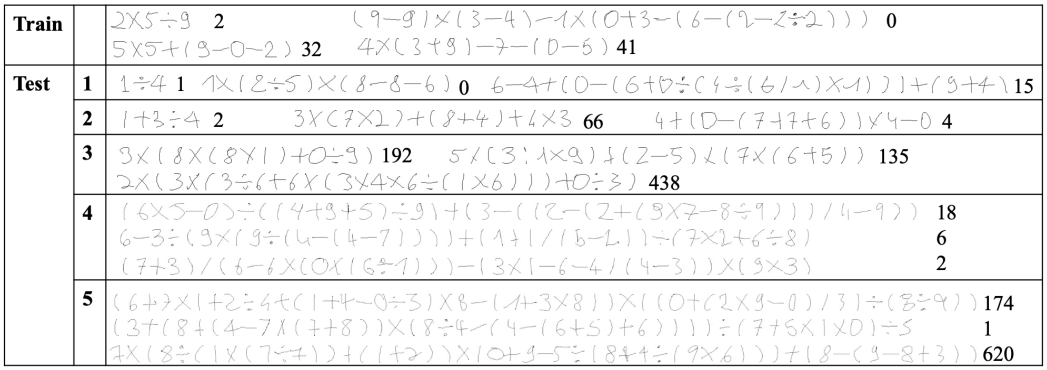


Figure S2: **Randomly selected examples from the training set and each subset of the test set.**

From a maximum likelihood prospective, the learning objective is to maximize the observed-data log likelihood $L(x, y) = \log p(y|x)$. Take the derivative of $L$ w.r.t. $\theta_p$,

$$\begin{aligned}\nabla_{\theta_p} L(x,y) &= \nabla_{\theta_p} \log p(y|x) = \frac{1}{p(y|x)} \nabla_{\theta_p} p(y|x) \\ &= \sum_{s,pt,et} \frac{p(s,pt,et,y|x;\Theta)}{\sum_{s',pt',et'} p(s',pt',et',y|x;\Theta)} \nabla_{\theta_p} \log p(s|x;\theta_p) \\ &= \mathbb{E}_{s,pt,et\sim p(s,pt,et|x,y)}[\nabla_{\theta_p} \log p(s|x;\theta_p)].\end{aligned} \tag{2}$$

Similarly, for $\theta_s, \theta_l$, we have

$$\nabla_{\theta_s} L(x,y) = \mathbb{E}_{s,pt,et\sim p(s,pt,et|x,y)}[\nabla_{\theta_s} \log p(pt|s;\theta_s)] \tag{3}$$

$$\nabla_{\theta_l} L(x,y) = \mathbb{E}_{s,pt,et\sim p(s,pt,et|x,y)}[\nabla_{\theta_l} \log p(et|pt;\theta_l)]. \tag{4}$$

where $p(s, pt, et|x, y)$ is the posterior distribution of $(s, pt, et)$ given $(x, y)$. Since $p(y|et)$ can only be 0 or 1, $p(s, pt, et|x, y)$ can be rewritten as:

$$p(s,pt,et|x,y) = \frac{p(s,pt,et,y|x;\Theta)}{\sum_{s',pt',et'} p(s',pt',et',y|x;\Theta)} = \begin{cases} 0, & \text{for } s,pt,et \notin Q \\ \frac{p(s,pt,et|x;\Theta)}{\sum_{s',pt',et'\in Q} p(s',pt',et'|x;\Theta)}, & \text{for } s,pt,et \in Q \end{cases} \tag{5}$$

where $Q = \{(s, pt, et) : p(y|et) = 1, s \in \Omega_s, pt \in \Omega_t, et \in \Omega_e\}$ is the set of $(s, pt, et)$ that generates $y$. Usually, $Q$ is a very small subset of the entire space of $(s, pt, et)$, *i.e.*, $Q \subseteq \Omega_s \times \Omega_t \times \Omega_e$, where $\times$ denotes the Cartesian product.

Since taking expectation w.r.t. this posterior distribution is intractable, we use Monte Carlo sampling to approximate it. Therefore, the learning procedure for an example $(x, y)$ can be depicted as following:

1. sample $\hat{s}, \hat{pt}, \hat{et} \sim p(s, pt, et|x, y)$;
2. use $(x, \hat{s})$ to update the perception model ($\theta_p$);
3. use $(\hat{s}, \hat{pt})$ to update the parsing model ($\theta_s$);
4. use $(\hat{pt}, \hat{et})$ to update the reasoning model ($\theta_l$).

### B.2 Arithmetic Neural-Symbolic (ANS) Model

The general framework of the desired neural-symbolic system described above is agnostic to the choice of functions and algorithms. Below we delineate a learnable implementation, named ANS, capable of learning generalizable concepts in arithmetic on the proposed HINT dataset.

#### B.2.1 Perception: Neural Network

The role of the perception module is to map a handwritten expression $x$ into a symbolic expression $s$. Since disentangling visual symbols from handwritten expressions is trivial in this domain[2], we

[2]Perfect disentanglement can be achieved by state-of-the-art unsupervised disentanglement learning methods (Burgess et al., 2019; Locatello et al., 2020)

assume the input as a sequence of handwritten images, where each image contains one symbol. We adopt a standard ResNet-18 (He et al., 2016) as the perception module to map each handwritten image into a probability distribution over the concept space $\Sigma$. Formally,

$$p(s|x;\theta_p) = \prod_i p(w_i|x_i;\theta_p) = \prod_i \texttt{softmax}(\phi(w_i, x_i;\theta_p)), \tag{6}$$

where $\phi(s, x;\theta_p)$ is a scoring function parameterized by a Neural Network (NN) with parameters $\theta_p$. Since learning such an NN from scratch is prohibitively challenging, the ResNet-18 is pre-trained unsupervisedly (Van Gansbeke et al., 2020) on unlabeled handwritten images.

### B.2.2 SYNTAX: DEPENDENCY PARSING

In our dependency parser, a *state* $c = (\alpha, \beta, A)$ consists of a *stack* $\alpha$, a *buffer* $\beta$, and a set of *dependency arcs* $A$. The initial state for a sequence $s = w_0 w_1 ... w_n$ is $\alpha = [\texttt{Root}], \beta = [w_0 w_1 ... w_n], A = \varnothing$. A state is regarded as terminal if the buffer is empty *and* the stack only contains the node `Root`. The parse tree can be derived from the dependency arcs $A$. Let $\alpha_i$ denote the $i$-th top element on the stack, and $\beta_i$ the $i$-th element on the buffer. The parser defines three types of transitions between states:

- LEFT-ARC: add an arc $\alpha_1 \rightarrow \alpha_2$ to $A$ and remove $\alpha_2$ from the stack $\alpha$. Precondition: $|\alpha| \geqslant 2$.
- RIGHT-ARC: add an arc $\alpha_2 \rightarrow \alpha_1$ to $A$ and remove $\alpha_1$ from the stack $\alpha$. Precondition: $|\alpha| \geqslant 2$.
- SHIFT: move $\beta_1$ from the buffer $\beta$ to the stack $\alpha$. Precondition: $|\beta| \geqslant 1$.

The goal of the parser is to predict a transition sequence from an initial state to a terminal state. As the parser is greedy, it attempts to predict one transition from $\mathcal{T} = \{\text{LEFT-ARC}, \text{RIGHT-ARC}, \text{SHIFT}\}$ at a time, based on the current state $c = (\alpha, \beta, A)$. The features for a state $c$ contains following three elements: (i) The top three words on the stack and buffer: $\alpha_i, \beta_i, i = 1, 2, 3$; (ii) The first and second leftmost/rightmost children of the top two words on the stack: $lc_1(\alpha_i), rc_1(\alpha_i), lc_2(\alpha_i), rc_2(\alpha_i), i = 1, 2$; (iii) The leftmost of leftmost/rightmost of rightmost children of the top two words on the stack: $lc_1(lc_1(\alpha_i)), rc_1(rc_1(\alpha_i)), i = 1, 2$. We use a special `Null` token for non-existent elements. Each element in the state representation is embedded to a $d$-dimensional vector $e \in R^d$, and the full embedding matrix is denoted as $E \in R^{|\Sigma| \times d}$, where $\Sigma$ is the concept space. The embedding vectors for all elements in the state are concatenated as its representation: $c = [e_1\ e_2 ... e_n] \in R^{nd}$. Given the state representation, we adopt a two-layer feed-forward NN to predict a transition.

$$h = \texttt{RELU}(W_1 c + b_1) \tag{7}$$

$$p = \texttt{softmax}(W_2 h + b_2), \tag{8}$$

where $W_1 \in R^{d_h \times nd}, b_1 \in R^{d_h}, W_2 \in R^{|\mathcal{T}| \times d_h}, b_2 \in R^{|\mathcal{T}|}$ are the weights and bias vectors in the NN and $d_h$ is the dimension of the hidden layer.

### B.2.3 SEMANTICS: PROGRAM SYNTHESIS

The semantics of concepts in HINT, including digits, operators, and parentheses, are all represented as programs composed from these primitives $L = \{\texttt{0}, \texttt{inc}, \texttt{dec}, \texttt{if}, \texttt{Y}\}$. During inference, these programs are used for reasoning to obtain the results. The learning for a concept $c$ is to find a program $\rho_c$ to maximize the following objective:

$$\rho_c = \arg\max_{\rho} p(\rho|D_c, L) \propto (D_c|\rho)\ p(\rho|L), \tag{9}$$

where $D_c$ denotes the input-output pairs of the concept $c$ for program induction, $p(D_c|\rho)$ the likelihood of the program $\rho$ explaining $D_c$, and $p(\rho|L)$ the prior of $\rho$ under the library $L$, which defines a generative model over programs. The maximization in Eq. (9) is achieved by a stochastic search process guided by a neural network, which is trained to approximate the posterior distribution $p(\rho|D_c, L)$.

### B.2.4 LEARNING BY DEDUCTION-ABDUCTION

In Appendix B.1, we derive a general learning procedure for such a neural-symbolic system. The key is to perform efficient sampling from the posterior distribution $p(s, pt, et|x, y)$. Algorithm 1 provides an overview of the proposed learning algorithm. In short, we generalize the back-search

algorithm in (Li et al., 2020a) to a *deduction-abduction* strategy to enable efficient sampling from the posterior distribution of perception, syntax, and semantics.

**Deduction** For a given example $(x, y)$, we first perform greedy deduction from $x$ to obtain a candidate solution of a compound tree $ct = (x, \hat{s}, \hat{pt}, \hat{et})$. This process is likely to produce a wrong result, thus requiring a separate abduction process to further correct it, detailed below.

**Abduction** To find a revised solution $ct^*$ that can reach the goal $y$, we search the neighbors of $ct$ in a top-down manner by performing abduction over perception ($s$), syntax ($pt$), and semantics ($et$), as detailed in Algorithm 2 and illustrated in Fig. 2. Our abduction strategy generalizes the perception-only, one-step back-search algorithm described in Li et al. 2020a to all three levels. The SOLVE function and the priority used in the top-down search are similarly to the ones in Li et al. 2020a. The abduction can also be extended to multiple steps, but we only use one step for lower computation overhead. The above deduction-abduction strategy likely behaves as a Metropolis-Hastings sampler for the posterior distribution (Li et al., 2020a).

Fig. S3 visualizes a concrete example illustrating the proposed deduction-abduction strategy in ANS.

**Algorithm 1:** Learning by Deduction-Abduction

**Input**: Training set $D = \{(x_i, y_i) : i = 1, 2, ..., N\}$
**Initial Module**: perception $\theta_p^{(0)}$, syntax $\theta_s^{(0)}$, semantics $\theta_l^{(0)}$
**for** $t \leftarrow 0$ to $T$ **do**
  Buffer $\mathcal{B} = \varnothing$
  **for** $(x, y) \in D$ **do**
    $ct = \text{DEDUCE}(x, \theta_p^{(t)}, \theta_s^{(t)}, \theta_l^{(t)})$
    $ct^* = \text{ABDUCE}(ct, y)$
    $\mathcal{B} = \mathcal{B} \cup \{ct^*\}$
  **end for**
  $\theta_p^{(t+1)}, \theta_s^{(t+1)}, \theta_l^{(t+1)} = \text{learn}(\mathcal{B}, \theta_p^{(t)}, \theta_s^{(t)}, \theta_l^{(t)})$
**end for**
**return** $\theta_p^{(T)}, \theta_s^{(T)}, \theta_l^{(T)}$
**function** DEDUCE($x, \theta_p, \theta_s, \theta_l$)
  sample $\hat{s} \sim p(s|x; \theta_p), \hat{pt} \sim p(pt|\hat{s}; \theta_s), \hat{et} = f(\hat{pt}; \theta_l)$
  **return** $ct = (x, \hat{s}, \hat{pt}, \hat{et})$
**end function**

**Algorithm 2:** Abduction

**function** ABDUCE($ct, y$)
  Q=PriorityQueue()
  Q.push(root($ct$), $y$, 1.0)
  **while** $A, y_A, p$ = Q.pop() **do**
    $A = (i, w, v, arcs)$ ▷ *(image, symbol, value, arcs)*
    **if** $A.v == y_A$ **then**
      **return** $A$
    **end if**
    ▷ *Abduce perception*
    **for** $w' \in \Sigma$ **do**
      $A' = A(w \rightarrow w')$
      **if** $A'.v == y_A$ **then**
        Q.push($A', y_A, p(A')$)
      **end if**
    **end for**
    ▷ *Abduce syntax*
    **for** $arc \in arcs$ **do**
      $A' = \text{rotate}(A, arc)$
      **if** $A'.v == y_A$ **then**
        Q.push($A', y_A, p(A')$)
      **end if**
    **end for**
    ▷ *Abduce semantics*
    $A' = A(v \rightarrow y_A)$
    Q.push($A', y_A, p(A')$)
    ▷ *Top-down search*
    **for** $B \in$ children($A$) **do**
      $y_B = \text{SOLVE}(B, A, y_A | \theta_l(A.w))$
      Q.push($B, y_B, p(B)$)
    **end for**
  **end while**
**end function**

## C EXPERIMENTAL SETUP

**Models** Both the ResNet-18 and the dependency parser in the proposed ANS model are trained by an Adam optimizer (Kingma & Ba, 2015) with a learning rate of $10^{-4}$ and a batch size of 512. The ResNet-18 is pre-trained unsupervisedly (Van Gansbeke et al., 2020) on unlabeled handwritten images extracted from the training set. In the dependency parser, the token embeddings have a

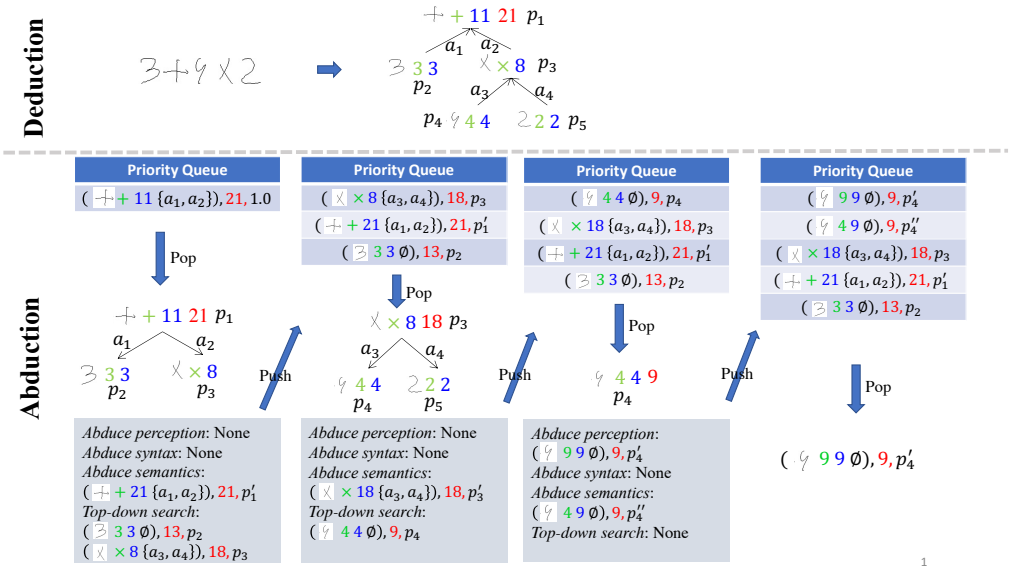



Figure S3: **An illustration of the deduction-abduction strategy in ANS.** Given a handwritten expression, the system first performs a greedy deduction to propose an initial solution, which generates a wrong result. In abduction, the root node, paired with the ground-truth result, is first pushed to the priority queue. The abduction over perception, syntax, and semantics is performed on the popped node to generate possible revisions. A top-down search is also applied to propagate the expected value to its children. All possible revisions are then pushed into the priority queue. This process is repeated until we find the most likely revision for the initial solution.

dimension of 50, and the hidden dimension of the transition classifier is 200. The program synthesis module is adapted from DreamCoder[3]. The three modules of ANS are jointly trained.

For end-to-end NN baselines, the task of HINT is formulated as a sequence-to-sequence problem: The input is an expression sequence, and the output is a sequence of digits, which is then converted to an integer as the predicted result. We test two popular seq2seq models: (1) BiGRU: the encoder is a bi-directional GRU (Chung et al., 2014) with three layers, and the decoder is a one-layer GRU, the token embeddings have a dimension of 128, and the hidden dimensions for the encoder and decoder are 128 and 256, respectively; (2) TRAN: a Transformer model (Vaswani et al., 2017) with three encoder-layers, three decoder-layers, and four attention heads for each layer, and the hidden dimension is 128. Before being fed into these models, the handwritten expressions are processed by the same ResNet-18 used in ANS. We test models with varied numbers of layers and report ones with the best results.

**Training** All models are trained for 100 epochs. To speed up the convergence, the training is guided by a simple curriculum from short expressions to long ones:

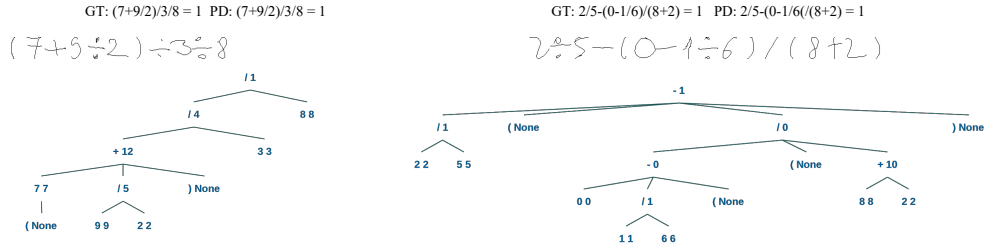
1. Epoch 0 $\sim$ 20: max length = 3
2. Epoch 20 $\sim$ 40: max length = 7
3. Epoch 40 $\sim$ 60: max length = 11
4. Epoch 60 $\sim$ 80: max length = 15
5. Epoch 80 $\sim$ 100: max length = $\infty$

**Evaluation Metric** We evaluate the models with the accuracy of final results. Note that a predicted result is considered correct when it *exactly* equals to the ground-truth.

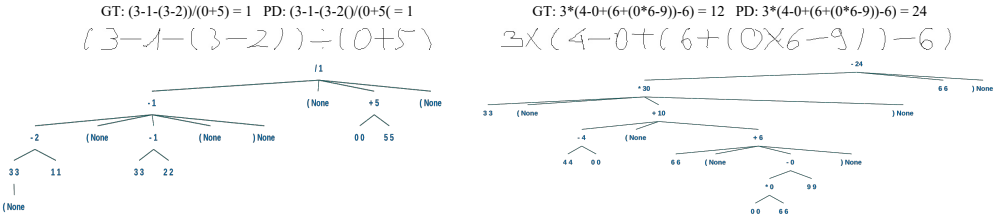
**Qualitative Examples** Fig. S4 shows several examples of the ANS predictions on each test subset.

[3] https://github.com/ellisk42/ec

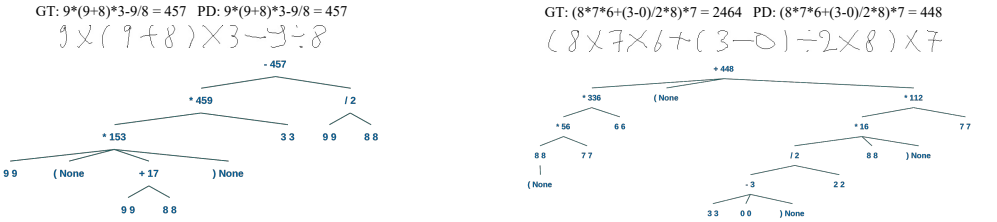
**Test subset 1**



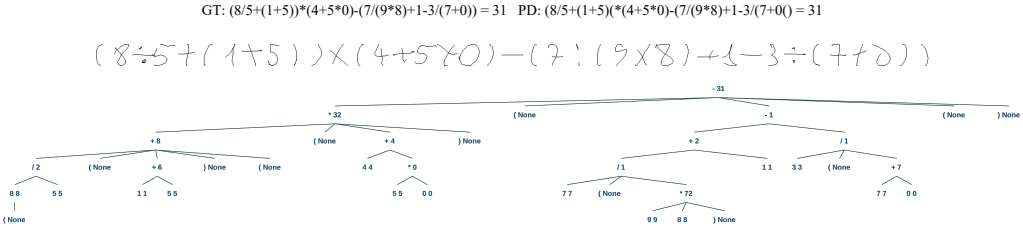

**Test subset 2**




**Test subset 3**




**Test subset 4**




**Test subset 5**

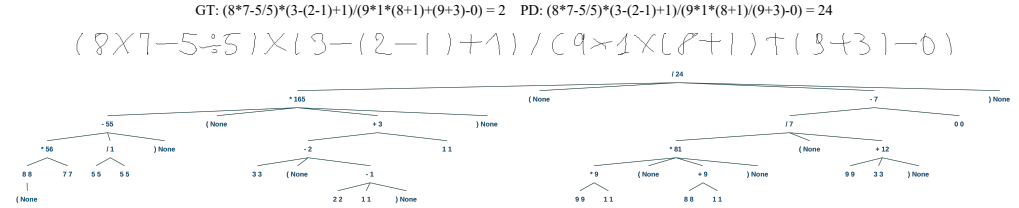



Figure S4: **Examples of ANS predictions on the test set.** "GT" and "PD" denote "ground-truth" and "prediction," respectively. Each node in the solution tree is a tuple of (symbol, value). Please check the attached codebase for more examples.

# D Related Work

## D.1 Three Levels of Concept Learning

The surge of deep neural networks (LeCun et al., 2015) in the last decade has significantly advanced the accuracy of **perception learning** from raw signals across multiple modalities, such as image

classification from image pixels (He et al., 2016; Krizhevsky et al., 2012) and automatic speech recognition from audio waveforms (Park et al., 2019; Hinton et al., 2012; Graves et al., 2013).

The goal of **syntax analysis** is to understand the compositional and recursive structures in various tasks, such as natural language parsing (Chen & Manning, 2014; Kitaev & Klein, 2018), image and video parsing (Tu et al., 2005; Zhu et al., 2007; Zhao & Zhu, 2011; Gupta et al., 2009; Qi et al., 2018a; Jia et al., 2020), scene understanding (Huang et al., 2018b;a; Qi et al., 2018b; Jiang et al., 2018; Chen et al., 2019; Yuan et al., 2020), task planning (Xie et al., 2018; Liu et al., 2018; Edmonds et al., 2019; Liu et al., 2019; Zhang et al., 2020b), and abstract reasoning (Zhang et al., 2019a;b; 2020a). There exist two major structural types: constituency structures (Kitaev & Klein, 2018) and dependency structures (Chen & Manning, 2014). Constituency structures use phrase structure grammar to organize input tokens into nested constituents, whereas dependency structures show which tokens depend on which other tokens.

**Semantics** of concepts essentially describe its causal effect. There are two primary semantic representations in symbolic reasoning. The first is *logic* (Lloyd, 2012; Manhaeve et al., 2018), which regards the semantic learning as inductive logic programming (Muggleton & De Raedt, 1994; Evans & Grefenstette, 2018)—a general framework to induce first-order logic theory from examples. The other representation is *program*, which treats the semantic learning as inductive program synthesis (Kulkarni et al., 2015; Lake et al., 2015; Balog et al., 2017; Devlin et al., 2017; Ellis et al., 2018a;b). Recently, Ellis et al. (2020) release a neural-guided program induction system, *DreamCoder*, which can efficiently discover interpretable, reusable, and generalizable knowledge across a wide range of domains.

However, aforementioned literature tackles *only one or two levels* of concept learning and usually requires *direct* supervision on model outputs. In contrast, in this paper we offer a more holistic perspective that addresses all three levels of concept learning, *i.e.*, perception, syntax, and semantics, taking one step closer to realize a versatile mechanism of concept learning under weak supervision. The design of three-level concept learning echoes a newly proposed challenge, HALMA, by Xie et al. (2021), but with a focus on perception instead of interaction with the environments.

### D.2 Systematic Generalization

The central question in systematic generalization is: How well can a learning agent perform in unseen scenarios given limited exposure to the underlying configurations (Grenander, 1993)? This question is also connected to the Language of Thought Hypothesis (Fodor, 1975): The systematicity, productivity, and inferential coherence characterize compositional generalization of concepts (Lake et al., 2015). As a prevailing property of human cognition, systematicity poses a central argument against connectionist models (Fodor et al., 1988). Recently, there have been several works to explore the systematic generalization of deep neural networks in different tasks (Lake & Baroni, 2018; Bahdanau et al., 2018; Keysers et al., 2019; Gordon et al., 2019; Xie et al., 2021). By going beyond traditional i.i.d. train/test split, the proposed HINT benchmark well-captures the characteristics of systematic generalization across different aspects of concepts w.r.t. perception, syntax, and semantics.

### D.3 Neural-Symbolic Integration

Researchers have proposed to combine statistical learning and symbolic reasoning, with pioneer efforts devoted to different directions, including representation learning and reasoning (Sun, 1994; Garcez et al., 2008; Manhaeve et al., 2018), abductive learning (Li et al., 2020a; Dai et al., 2019; Zhou, 2019), knowledge abstraction (Hinton et al., 2006; Bader et al., 2009), *etc*. There also have been recent works on the application of neural-symbolic methods, such as neural-symbolic visual reasoning and program synthesis (Yi et al., 2018; Mao et al., 2018; Li et al., 2020b; Parisotto et al., 2016), semantic parsing (Liang et al., 2016; Yin et al., 2018), and math word problems (Lample & Charton, 2020; Lee et al., 2020). Current neural-symbolic approaches often require a perfect domain-specific language, including both the syntax and semantics of the targeted domain. In comparison, the proposed model relaxes such a strict requirement and enables the learning of syntax and semantics.

## E DISCUSSION: CONTRIBUTIONS AND LIMITATIONS

In this paper, we take inspiration from how humans learn arithmetic and present a new challenge for the machine learning community, HINT, which serves as a minimal yet complete benchmark towards studying systematic generalization of concepts w.r.t. perception, syntax, and semantics. Additionally, we propose a neural-symbolic system, Arithmetic Neural-Symbolic (ANS), to approach this challenge. ANS integrates recent efforts from the disciplines of neural networks, grammar parsing, and program synthesis. One potential future work is to extend our model to other domains and applications.

**Extending to other domains.** To extend our model to other domains with varieties of semantics, such as visual reasoning (Johnson et al., 2017; Hudson & Manning, 2019) and question answering (Rajpurkar et al., 2016), we may consider to inject contexts into the semantics of concepts and capture their inherent stochastic nature with probabilistic programs (Ghahramani, 2015; Carpenter et al., 2017; Ge et al., 2018; Bingham et al., 2019; Holtzen et al., 2020).