# IMPLY - a Logic Learning Game for Pre-literate Children

Mario Wenzel[1,*,†]

[1]University of Halle-Wittenberg, Von-Seckendorff-Platz 1 / Kurt-Mothes-Str. 1, 06120, Halle, Germany

**Abstract**

This paper describes the logic training workspace "IMPLY - interactive models for pre-literate youth". IMPLY aims at pre-literate children by using images as literals for concepts or events in a small game universe. The literals are used in general propositional Horn clauses in implication form to describe the valid models, or worlds, of this specific game universe. Multiple game modes allow the exploration of the implication as a fundamental building block of logic programming.

**Keywords**

Logical Thinking, Educational Game, Logic Programming

## 1. Introduction

It has been well established that if-then-rules are a suitable formalism to describe computation [1] and have been used as a base for computer science education for children [2, 3]. More generally, Horn clauses are the basis of several computational models of logic programming, most notably Prolog, Datalog, and Answer Set Programming [4, 5].

Such rules take on the form of "A if B and C", "if B and C then A", "$A \leftarrow B \wedge C$", "`A :- B, C`", or similar. Because of the textual nature of the rules, literacy is presumed for the users of a specific rule language and its tools. Even when using visual programming languages that manage to minimise frustrations with syntax and syntax errors, like blockly-Prolog[1], blawx [6], or FOLL-E [7]: the rules take on 2D shapes, but their meaning is not conveyed through the shapes. The meaning is conveyed by having the shapes labeled "Rule", "Fact", "Body", etc. or, in the case of Blawx "If we know ... then we also know ...".

In the FOLL-E project the goal is to distinguish colorful robots via rules. And while the robots themselves (which represent models or valid/invalid robot configurations for the underlying rules) are easily graspable without being able to read, the rule shapes are labeled with their meaning, as they are not self-explanatory. And of course, the shapes do not need to be, if the child creating the rules can read.

Because of this dependency on literacy to convey the abstract reading of logic programs, primary education is typically argued as the earliest target for logic programming education [8].

To explore the question whether literacy is truly necessary to train the forward reasoning skills used in logic programming, we introduce the game "IMPLY - interactive models for pre-literate youth". IMPLY is a web-application that presents supervised exercises in a form that avoids written language where possible. The core of IMPLY are sets of if-then rules where the literals in the rules are images that are dragged onto the "if" or "then" area of the rules. Multiple game modes allow the player to explore the effects their rules have on the validity of worlds within the game universe.

In Section 2 we give an overview over the user interface of the game. In Section 3 we present the specific semantics of an IMPLY "program". In section 4 we explain the four different game modes. Section 5 describes the trial performed by this author and their 4 year old son. We note the drawn conclusions from the experiment in Section 6.

The main contribution of this paper is the IMPLY application as a discussion starter of when, at the earliest, we may start with the introduction of concepts pertaining to formal reasoning, and which

*Fourth Prolog Education Workshop 2026, July 19, 2026, Lisbon, Portugal*

mario.wenzel@informatik.uni-halle.de (M. Wenzel)

© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

[1]https://www.boles.de/programmierkurs-java/blocklyprolog/

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073
published 2026-11-03

CEUR-WS.org/Vol-4311/PEG_regular_02.pdf

elements of previous approaches need to be adapted, if possible, to accommodate children at pre-literacy ages. The application can be tried out at https://tech.maweki.de/files/imply/.

## 2. Game Overview

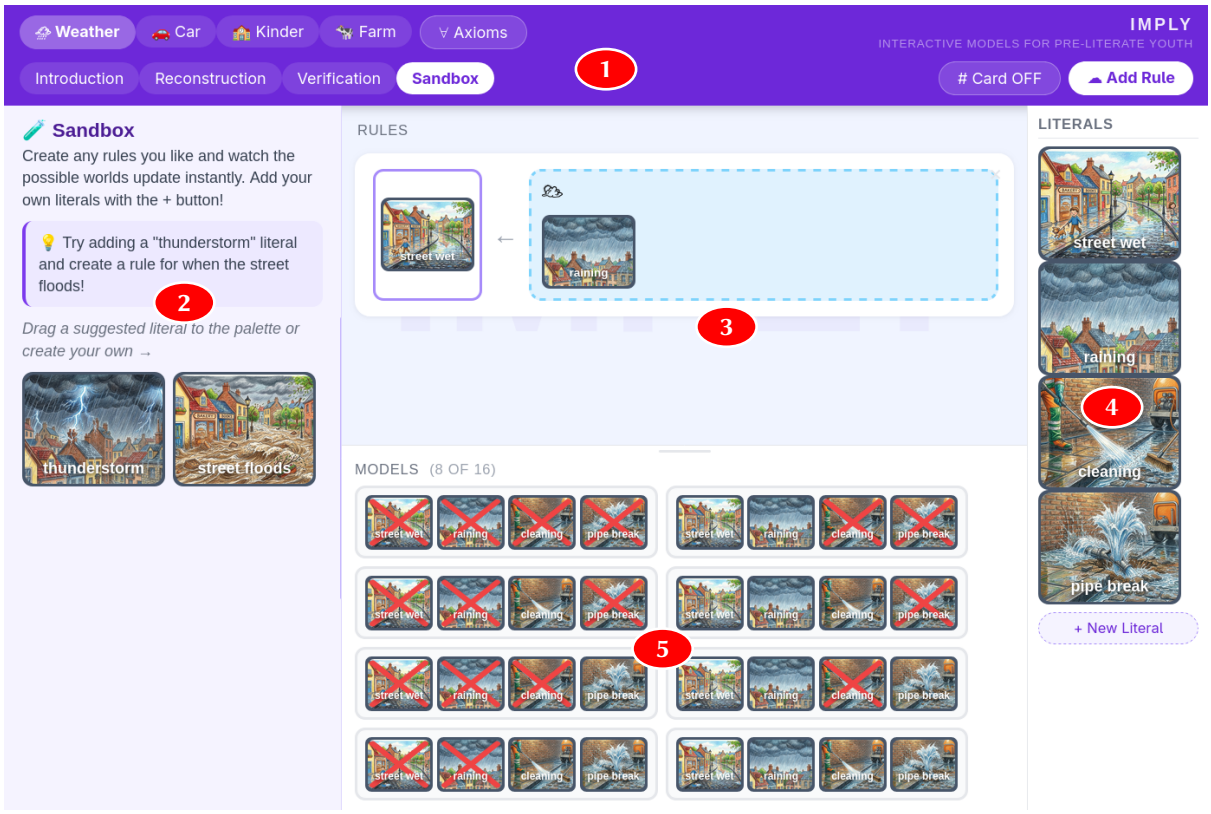



**Figure 1:** Overview of the user interface of IMPLY with highlighted areas of interest.

The game's user interface (UI) is split into five areas (see Figure 1).

1. The **title panel** shows the current game universe (a set of literals with a common theme and corresponding tasks), the current game mode (see Section 4), a toggle for enabling cardinality constraints (see Section 6), and a game-mode specific button which is usually used to add rules. The "Axioms" button shows the rules of our system of logic (see Figure 2), explaining the rule of excluded middle, the meaning of implication, the meaning of a conjunction, negation, and what a fact is. The explanation is in text form and it is there to guide non-professional supervisors.
2. The **explanation panel** introduces and explains different tasks depending on the chosen universe and game mode.
3. The **rule area** shows the current rules that are evaluated. A rule consists of a head-area (or "then-area") and a body-area (or "if-area") that are used as drop-targets for literals.
4. The **literals panel** shows the available literals that may be used to fulfill the task. In most game modes the literals are dragged to the rule area. Sandbox mode allows adding custom literals. In Challenge mode the literals may be "toggled" to select a certain valuation for them that is consistent with the rules. The literals may be clicked/tapped to show a larger picture and its name is read out using the browser's text-to-speech engine, allowing to recapitulate the meaning of the literals without supervision.
5. The **models/worlds panel** shows information about the models obtained from the specified rules. In reconstruction mode it shows which models/worlds are also in the set of target models/worlds.

And in verification mode the model view is initially empty, as the player has to generate worlds to test them against the rules.

## 3. Semantics

The rule area allows rules of the form $A \leftarrow B_1 \wedge \cdots \wedge B_n$, where $A$ and $B_i$ are either positive or negated literals that are taken from the literals panel. Rules with $n = 0$ are allowed and we call them facts.

The game starts with the simple "Weather" universe that consists of the literals "street (w)et", "(r)aining", "street (c)leaning", "pipe (b)reak", with the intended rules conveying that the street is wet if any of the other literals are true, i.e. $P = \{W \leftarrow R, W \leftarrow C, W \leftarrow B\}$.

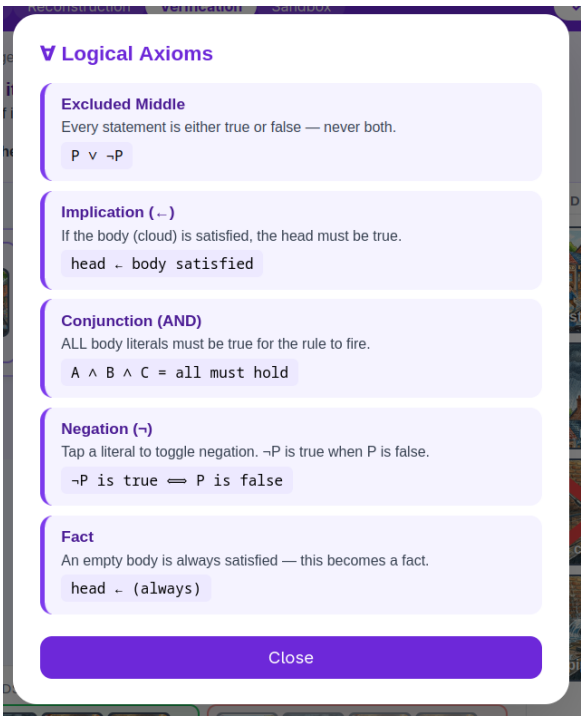



Figure 2: Axioms window.

**Standard Model Semantics:** In the initial development we checked all possible valuations of literals and marked all worlds (models) as valid where all the rules are true. This also included the world where the street is wet and none of the other literals are true (world $W\overline{RCB}$). This strategy is an open world view of the program, as there may be a myriad of other reasons for the street being wet that we just have not modelled. But this was deemed counter-intuitive with regards to the target audience, hence we aim for a closed world view. We characterize the valid worlds for the set of literals $L_P$ of the program $P$ as $\{W \mid W \in L_P^2 \wedge W \models P\}$.

**Minimal Model Semantics:** One solution would be to check for minimality, as the world $\overline{WRCB}$ is also valid, but then this would be, in the absence of any facts (or negation), the only valid world. Taking a hint from Datalog literature [9], we separate the literals into "intensional" literals, the ones appearing in the head of a rule, and "extensional" literals - the other ones. The extensional literals form the "database" for our logic program and we want to obtain the minimal models for all possible extensional databases via the standard consequence operator $\mathsf{T}_P$. In other words, we execute our logic program for all valuations of the extensional literals in order to obtain values for the intensional literals. We characterize the valid worlds for the set of extensional literals $E_P$ of the program $P$ as $\{lfp(\mathsf{T}_{P \cup E'}) \mid E' \in E_P^2\}$.

**Fixpoint Model Semantics:** In the minimal model semantics the ruleset $P = \{W \leftarrow R, R \leftarrow W, C \leftarrow B, B \leftarrow C\}$ has no extensional literals (as all literals appear in rule heads) and therefore the only valid world is $\overline{WRCB}$. Even though, again with regards to the target audience, it might be more intuitive to have $WRCB$, $\overline{W}RCB$, $WR\overline{C}B$, and $\overline{WRCB}$ as valid worlds. So in this case it might be useful to check all worlds whether they are a fixpoint w.r.t. to the consequence operator $\mathsf{T}_P$, while also trying out all possible databases. This basically allows for circular reasoning.
We characterize the valid worlds for the set of literals $L_P$ and the set of extensional literals $E_P$ of the program $P$ as $\{W \mid W \in L_P^2 \wedge W \models P \wedge \mathsf{T}_{P \cup (W \cap E_P)}(W)) = W\}$.

Though not implemented, denials of the form $\bot \leftarrow B_1 \wedge \cdots \wedge B_n$ are useful across all considered semantics as a means of excluding undesirable worlds. Their role becomes particularly important under standard and fixpoint model semantics, where they compensate for the permissiveness by explicitly

ruling out unintuitive configurations.

## 4. Game Modes

In the title panel the player can select one of four game modes:

**Introduction:** This mode shows some details of the universe and what kind of rules we expect ("When it rains, the street gets wet. If it is not raining, the street stays dry."), together with a call to action, in the explanation panel. To check the player's work, some implications are added below the introductory text matching the text ($R \to W, \overline{R} \to \overline{W}$). The goal for the player is to add a set of rules so that all valid worlds match the expected implications. This may be as simple as copying a rule exactly, or it may be more complex.

In the farm universe where plants only grow if it is sunny and it has rained (and the farmer is happy if plants grow), the implications given are "if it's not sunny and it rained, plants don't grow", "if it's sunny and it hasn't rained, plants don't grow" (and "if it has rained, and it is sunny, and plants grow, the farmer is happy"). Here the expectation is, that the added rule is "if it's sunny and it has rained, the plants grow".

The player should note how the set of valid worlds changes in the worlds panel when they add or change rules.

**Reconstruction:** This mode is about rule synthesis as well. Though instead of example implications, the full set of valid worlds is shown in the explanation panel (generated by a master solution of rules). The player should add rules to cover the given worlds exactly. In the world panel all worlds for the player's rules are shown and marked with ✓or ✗, denoting whether they are correctly or incorrectly marked positive (true and false positives). If there are worlds we wrongfully dismissed (false negatives), the worlds panel also shows "X missing".

This game mode is similar to FOLL-E's game mode where examples for positive and negative robot configurations are shown and it is the player's goal to introduce a rule that correctly distinguishes both sets.

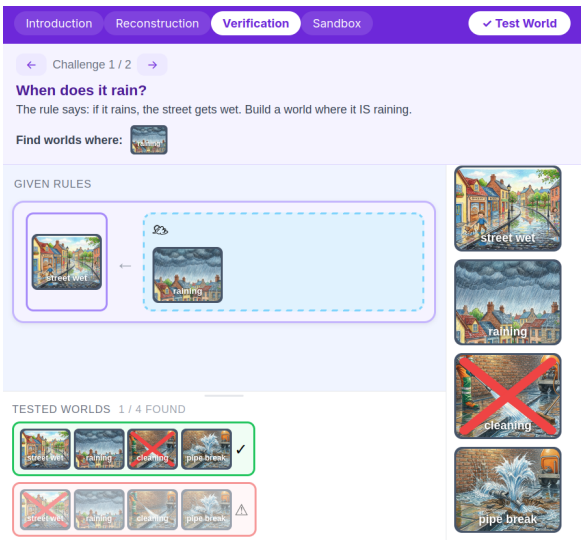



Figure 3: Verification game mode.

**Verification:** In this mode the player uses the literals panel to select a valuation for the literals. The rule area has rules that may or may not be ones from the previous modes, while the explanation panel provides an additional constraint for the worlds ("Build all worlds where it is raining.", see Figure 3). The player's goal is to find all worlds that are valid worlds for the given rules and that fit the additional constraint. Whether a world is correct or not is shown in the worlds panel, where each tested world is added and marked as correct or incorrect. A text shows "X / Y WORLDS FOUND".

**Sandbox:** The player is now allowed to add more literals to the program via the literals panel (see Figure 1). The explanation panel contains suggestions on what to model next ("Try adding a 'thunderstorm' literal and create a rule for when the street floods!"), as well as corresponding example literals ("thunderstorm" and "street floods"), which may be dragged to the literals panel. The models panel shows all possible worlds for the defined literals and rules. This list might be exponentially long in the number of literals, which may get unwieldy. The player's work is not checked, but they may explore the program and the resulting worlds at will.

## 5. Trial

The game has been tried out using the author's 4yr old child (subsequently "the child") as a test subject. While sitting on the author's lap, the semantics of a rule (if and then) and how they can be created was explained. For each game mode a brief introduction was given and any introductory text read. The following observations were made:

- The child had no problem understanding the meaning of the literals and how to use them in rules.
- The if- and then-areas of the rules were not easily distinguished and the child asked multiple times to be reminded of which area was which.
- Dragging and dropping literals between the literals panel and the rules area was intuitive.
- The child could easily recreate the rules from the introductory mode and was interested and engaged by the changes in the explanation and worlds panel.
- The reconstruction mode was too difficult as the child lost the overview over all the valid worlds it was supposed to capture. Even if a rule was tentatively established and the worlds panel showed the evaluation, the child had no idea how to repair a wrong rule, or how to continue synthesizing rules to capture more valid worlds. Without help it was not clear for the child, how to (systematically) approach the task.
- The reconstruction task was easily accomplished, as the child understood the rules and was able to enumerate all the valid worlds systematically.
- The child easily understood the suggested task in sandbox mode and added the literals from the explanation panel to the literals panel. The child created rules, was able to explain them, and observed the valid worlds. The created rules were not always consistent with the real world though. This might be due to lack of knowledge about causes and effects in the real world.

## 6. Notes and Conclusion

IMPLY is a first step towards training forward reasoning skills in pre-literate children, skills that later play an important role in logic programming. The (very small) trial with the author's child suggests that the basic mechanics of the game are intuitive: the child was able to understand the meaning of literals, the use of images as representations, and implication-based rules. However, some game modes proved too difficult, indicating that additional simplification or scaffolding may be necessary for certain tasks. For example, the reconstruction mode might be simplified by showing only a subset of the valid worlds (and invalid worlds, as in FOLL-E) at a time, or by providing hints on how rules could be modified to capture additional valid worlds. An evaluation with a larger and more diverse group of test subjects could be of value.

A central design decision concerns the intended meaning of implication in the game worlds. In the current universes, implication is interpreted causally. For example, in the weather universe, the intended reading is that the street becomes wet if it rains. Such causal readings appear intuitive for the target audience, provided that children have developed some understanding of causal relations and of how the depicted events in a universe relate to one another. It may even be sufficient that children merely recognize that events can stand in causal relationships, since worlds generated by "wrong" rules can still be explored and compared. In either case, sensitivity to causal relations develops very early, classically placed in the second year of development [10], with some studies suggesting understanding of physical or mechanical causality as early as six months [11].

If we compare the causal reading "If ... happens, then ... happens" of IMPLY to the "If we know ... then we also know ..." reading of blawx, the simplification and limitations become clear and game universes need to be designed around such simple readings of the rules. Creating more universes with different readings for the implication could make it clearer what is easy and what is difficult for that age group. Though for some readings of the implication it could be more difficult to create appropriate imagery. It is also unclear whether it is better for the player if the game uses images that show more abstract or more concrete situations and whether this makes a difference in understanding.

Another important question is the choice of semantics. Great care must be taken to decide on semantics that are suitable and intuitive for the target audience. Standard model semantics may be too permissive, while minimal model semantics may in some cases be too restrictive. Fixpoint semantics are more permissive than minimal model semantics, but potentially harder to explain and computationally more demanding, since all fixpoints for all databases of a program must be found rather than a unique minimal one per database. On the one hand, semantics should be chosen according to the intended learning outcomes and the children's prior intuitions. On the other hand, it may be preferable to restrict gameplay to programs for which the relevant semantics coincide, thereby avoiding distinctions that are likely too complex for the intended audience.
It might be useful using the verification game mode to find out what the target audience actually believes the semantics to be, specifically concerning the circular reasoning allowed by the fixpoint semantics. "I am no longer their friend because they are not my friend" is sometimes heard in the author's household. Do we expclitly want to exclude circular reasoning because it's not the semantics we want to convey and use later on, even though children might intuitionally find these models?

At present, all literals in IMPLY are parameter free. This simplification keeps the mechanics intuitive, but also limits expressiveness and the complexity of the resulting worlds. Future work could investigate how parameters may be introduced while remaining accessible for pre-literate children. For example, constants could be represented by concrete shapes, colors, or other visual cues, while shared variables might be represented by hooks tied together with a virtual string. If parameters are introduced, the number of possible databases grows rapidly, raising further questions: how can many worlds be represented, is full enumeration still feasible, or are abstraction and summarization required? And if so, which are suitable for the target audience?

Expanding the game language with cardinality constraints is another promising direction, and a prototype is already included in IMPLY. With a button in the title panel, cardinality constraints can be enabled; instead of pure conjunction, rule bodies then allow the player to specify a minimum number of literals that must hold. This makes it possible to express conjunction ("at least $n$ of $n$") as well as disjunction ("at least 1 of $n$"). For example, the introductory weather rule can be shortened to $W \leftarrow 1$ to 3 of $\{R, C, B\}$. Since counting is also trained early in childhood, it may be worthwhile to investigate whether cardinality constraints provide a more natural route to conjunction and disjunction.

Beyond the two current core game modes—**reconstruction** and **verification**—many additional logic-programming tasks appear suitable for gamification. Possible extensions include **instantiation**, where players search for suitable rule instances to derive facts bottom-up; **provenance**, where a Prolog-style top-down reading of rules is used to build proof trees for target literals; **database synthesis**, where valuations for extensional literals must be chosen to achieve a desired valuation of intensional literals; and **minimisation**, where players identify and remove superfluous rules or facts implied by the remainder of the program.

Overall, IMPLY suggests that fundamental ideas from logic programming can be translated into intuitive visual interactions for very young learners. While substantial design questions remain regarding scaffolding, semantics, and scalability, the prototype indicates that pre-literate children may already be able to engage meaningfully with implication, models, and rule-based reasoning.

## Declaration on Generative AI

The IMPLY application presented in this paper was developed with the support of generative AI: Claude (Anthropic) was used to generate application code, and Google Gemini was used to generate the images that represent the literals of the game universes. These images are part of the application and therefore appear in the application screenshots in Figures 1, 2, and 3. Parts of the application code were generated using drafts of this paper as the specification. During the preparation of this work, the author used Claude in order to: Formatting assistance. After using these tools/services, the author reviewed and edited the content as needed and takes full responsibility for the publication's content.

## References


[1] R. A. Kowalski, Algorithm = logic + control, Commun. ACM 22 (1979) 424–436. URL: https://doi.org/10.1145/359131.359136. doi:10.1145/359131.359136.

[2] R. A. Kowalski, Logic as a computer language for children, in: 5th European Conference on Artificial Intelligence, ECAI 1982, Paris, 1982, Proceedings, 1982, pp. 2–10.

[3] R. Ennals, Teaching logic as a computer language in schools, in: M. V. Caneghem (Ed.), Proceedings of the First International Logic Programming Conference, Faculté des Science de Luminy, ADDP-GIA, Marseille, France, September, 14-17, 1982, ADDP-GIA, 1982, pp. 99–104.

[4] J. W. Lloyd, Foundations of Logic Programming, 2nd Edition, Springer, 1987. URL: https://doi.org/10.1007/978-3-642-83189-8. doi:10.1007/978-3-642-83189-8.

[5] V. Lifschitz, Answer Set Programming, Springer, 2019. URL: https://doi.org/10.1007/978-3-030-24658-7. doi:10.1007/978-3-030-24658-7.

[6] J. Morris, Blawx: Web-based user-friendly rules as code, in: J. Arias, R. Calegari, L. Dickens, W. Faber, J. Fandinno, G. Gupta, M. Hecher, D. Inclezan, E. LeBlanc, M. Morak, E. Salazar, J. Zangari (Eds.), Proceedings of the International Conference on Logic Programming 2022 Workshops co-located with the 38th International Conference on Logic Programming (ICLP 2022), Haifa, Israel, July 31st - August 1st, 2022, CEUR Workshop Proceedings, CEUR-WS.org, 2022. URL: https://ceur-ws.org/Vol-3193/paper4GDE.pdf.

[7] S. Vandevelde, J. Vennekens, FOLL-E: teaching first order logic to children, in: B. Williams, Y. Chen, J. Neville (Eds.), Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023, AAAI Press, 2023, pp. 15869–15876. URL: https://doi.org/10.1609/aaai.v37i13.26884. doi:10.1609/AAAI.V37I13.26884.

[8] L. A. Cecchi, J. P. Rodriguez, V. Dahl, Logic programming at elementary school: Why, what and how should we teach logic programming to children?, in: D. S. Warren, V. Dahl, T. Eiter, M. V. Hermenegildo, R. A. Kowalski, F. Rossi (Eds.), Prolog: The Next 50 Years, volume 13900 of *Lecture Notes in Computer Science*, Springer, 2023, pp. 131–143. URL: https://doi.org/10.1007/978-3-031-35254-6_11. doi:10.1007/978-3-031-35254-6_11.

[9] S. Ceri, G. Gottlob, L. Tanca, Logic Programming and Databases, Surveys in computer science, Springer, 1990. URL: http://www.worldcat.org/oclc/20595273.

[10] J. I. B. Piaget, The construction of reality in the child, 1954. URL: https://api.semanticscholar.org/CorpusID:20670356.

[11] A. M. Leslie, S. Keeble, Do six-month-old infants perceive causality?, Cognition 25 (1987) 265–288. URL: https://www.sciencedirect.com/science/article/pii/S0010027787800069. doi:https://doi.org/10.1016/S0010-0277(87)80006-9.