New Ideas in Psychology 61 (2021) 100838

Contents lists available at ScienceDirect

New Ideas in Psychology

journal homepage: http://www.elsevier.com/locate/newideapsych

Using logic programming for theory representation and scientific inference☆

Jean-Christophe Rohner[a,*], Håkan Kjellerstrand[b]

[a] Department of Psychology, Lund University, Sweden
[b] Independent Researcher, Malmö, Sweden

ARTICLE INFO

Keywords:
Logic programming
Prolog
Scientific inference
Scientific rigor

ABSTRACT

The aim of this paper is to show that logic programming is a powerful tool for representing scientific theories and for scientific inference. In a logic program it is possible to encode the qualitative and quantitative components of a theory in first order predicate logic, which is a highly expressive formal language. A theory program can then be handed to an algorithm that reasons about the theory. We discuss the advantages of logic programming with regard to building formal theories and present a novel software package for scientific inference: Theory Toolbox. Theory Toolbox can derive any conclusions that are entailed by a theory, explain why a certain conclusion follows from a theory, and evaluate a theory with regard to its internal coherence and generalizability. Because logic is, or should be, a cornerstone of scientific practice, we believe that our paper can make an important contribution to scientific psychology.

The aim of this paper is to show that logic programming (LP) is a powerful tool for representing scientific theories and for scientific inference.[1] By "theory" we mean a systematic qualitative and quantitative description of events, their relations and all background assumptions on which these relations depend. Theories describe, predict and explain relevant psychological phenomena, and they make definite (falsifiable) statements about the world. With LP it is possible to encode a theory in first order predicate logic. We intend to show that doing so has two important consequences. First, such a representation is highly expressive, meaning that it has the capacity to capture even intricate nuances of human thought, feeling and behavior. Secondly, such a representation can be handed to an inference algorithm that generates valid conclusions. A valid conclusion is a conclusion that is bound to be true if a theory and its background assumptions are true. So if a theory is a corroborated[2] one, LP can be used to make accurate predictions and deduce accurate explanations. If a theory is a provisional one, LP can be used to derive predictions for empirical testing, check whether the theory is internally coherent, and examine how falsifiable it is, among other things.

Logic has a long and revered history in philosophy but is also, sometimes colloquially, referred to as the ideal way of reasoning in other scientific disciplines (e.g. Shapiro & Kouri Kissel, 2018). To quote an introductory textbook on scientific methods in psychology "A scientific theory is a logically organized set of propositions that defines events, describes relationships among events, and explains the occurrence of events" (Shaughnessy et al., 2000, p. 49). With the advent of AI, logic also started to play an important role as a knowledge representation scheme: Standard textbooks in AI present first order logic as one of the most powerful formalisms for constructing intelligent agents that represent and reason about complex knowledge domains. Other knowledge representation languages, like the Web Ontology Language (OWL), are considered subsets of first order logic (Russell et al., 2010). In first order logic it is possible to describe qualitative and quantitative relations between singular objects and sets of objects; such relations can

☆ We want to thank the following people for their kind help during this project: Alan Baljeu, Anne Ogborn, Carlo Capelli, Daniel Lyons, David Tonhofer, Eric Taucher, Erik J Olsson, Fabrizio Riguzzi, @false, @Guy Coder, Jan Wielemaker, @lurker, Markus Triska, Martin Bäckström, @Mostowski Collapse, Paul Brown, Paulo Moura, Peter Ludemann, and @repeat.

* Corresponding author. BOX213, 221 00, Lund, Sweden.
E-mail address: jean-christophe.rohner@psy.lu.se (J.-C. Rohner).
URL: http://www.hakank.org (H. Kjellerstrand).

[1] We use the term "theory representation" instead of "knowledge representation", because the work in this paper applies to established theories as well as preliminary theories (which do not constitute knowledge in the sense of being justified true beliefs).

[2] In the sense of having survived multiple tests (Popper, 1972).

https://doi.org/10.1016/j.newideapsych.2020.100838
Received 4 June 2020; Received in revised form 21 September 2020; Accepted 17 November 2020
Available online 16 December 2020

have any degree of complexity (within reasonable limits). This means that it is possible to combine a representation of the abstract semantic content of a theory (traditionally stated in natural language) with a representation of the (abstract) mathematical content of the theory into a single formal system. This system can then be used as a basis for scientific inference.

Prolog is one of the most widely used first order LP languages (Russell et al., 2010). It was developed by Alain Colmerauer and Philippe Roussel in the 1970's (Colmerauer & Roussel, 1996). Some examples of real world settings where Prolog has been used are the International Space Station project (Rayner et al., 2005), the IBM Watson project (Lally & Fodor, 2011) and a widely used airline ticket booking system (Wilson, 2005). In Prolog it is possible to formalize a theory by writing down a set of definite clauses (we will return to what this means later). This clausal form logic is not full first order logic but a syntactic subset thereof. However, almost any formula in full first order logic can be rewritten in this clausal form (Russell et al., 2010). Semantically, Prolog thus retains much of the expressive power of full first order logic. In addition, Prolog is Turing complete; this means that it is possible to write logic programs that can perform *any* currently known computation (Crookes, 1988; Triska, 2020). A good introductory text to Prolog is *Programming in Prolog: Using the ISO Standard* (Clocksin & Mellish, 2012). More advanced texts have been written by Bratko (2001), Sterling and Shapiro (1994), and Triska (2020). In the current project we use Prolog to implement all logic programs.

Despite these apparent advantages, it seems that papers that describe LP, or first order logic, as theory representation tools in psychology are surprisingly rare. Our searches of Scopus for "first order logic", "predicate logic", "logic programming" and "Prolog" revealed 27, 17, 72 and 58 matches, respectively (restricted to the subject psychology, title, abstract and keywords, November 2020). As far as we could tell, none of these works contain a general and systematic discussion of the advantages of LP for representing scientific theories and for scientific inference in psychology. The papers that, relatively speaking, come closest to our aims, either use first order logic to model other things than scientific theories, e.g. common sense psychology (Gordon & Hobbs, 2017) and how subjects learn and think about causal relations (Griffiths & Tenenbaum, 2007); or use first order logic to model very specific phenomena, e.g. neural processes in schizophrenia (Flax, 2007), social interaction between primates (Bond, 1999), and a specific emotion theory (Adam et al., 2009). Given this apparent gap in the literature, and the scientific significance of logic, we believe that our paper can make a novel and important contribution to psychology.

Being able to encode theories into accurate representations, and being able to make correct inferences based on these representations, are essential components in any scientific project. Achieving these goals critically depends on having a good theory representation language at hand. We are not convinced that natural languages, such as English, are ideal for these purposes, however. In contrast to logic, natural languages are not well defined symbolic representations that are suitable for logical inference.[3] For example, it is very hard to design algorithms that reason deductively from typical English sentences (one and the same concept or relation can be described in very many ways). Also, natural language expressions tend to be ambiguous which can lead to erroneous conclusions. Mathematics, on the other hand, has important strengths when it comes to precision and validity, and such formalizations are indeed common in psychology. Well known examples, besides basic bivariate statistical models, include general linear models (e.g. Cohen et al., 2014), structural equation models (e.g. McArdle & Kadlec, 2013), multinomial processing tree models (Batchelder & Riefer, 1999) and Bayesian models (e.g. Bielza & Larrañaga, 2014). Nonetheless, we believe that LP can complement these schemes in an important way: By adding semantic content to formal representations of psychological theories. We will explain what this means more precisely in the remainder of this paper.

There is also a fundamental difference between LP, which is a type of symbolic AI, and what can be called "statistical AI", in terms of how objects, properties and relations are represented and inferred. Statistical AI consist of supervised and unsupervised machine learning algorithms that operate on large vector representations; some well-known examples are classical neural networks, deep neural networks and support vector machines. These are highly expressive formalisms that achieve good performance levels when it comes to applied prediction (e.g. Marwala, 2019). But even if some progress has been made recently (e.g. Murdoch et al., 2019), they are still inherently hard to understand by humans. Logic programs, in contrast, use words to represent objects, properties and relations; this makes them more similar to natural language, and therefore more transparent. As we will show later, it is also possible to follow all the steps that lead up to a conclusion from a logic program. Considering the importance of accountability in AI, these features of LP seem attractive (e.g. Jobin et al., 2019).

Logic programming can also be contrasted with a symbolic cognitive modeling approach (see Kriegeskorte & Douglas, 2018 for an overview). ACT-R is perhaps the most well-known example (Anderson et al., 2004). Even if a detailed comparison is beyond the present scope, there is an important difference that deserves some attention. Typically, cognitive models are special purpose models that are intended to be structurally and computationally similar to the mind (or brain). ACT-R, for example, has modules for perceptual and motor processes, procedural memory and declarative memory. In contrast, LP can be used to describe any kind of theory, as long as it can be written down in clausal form; it is up to the user to decide what concepts and relations it covers. Since Prolog is a Turing complete general purpose language, it can be used to represent such divergent things as, say, the mutagenicity of chemical molecules (Srinivasan et al., 1996), the rules of chess (Bratko, 2001), and a symbolic equation solver (Sterling et al., 1989).

This paper is structured around three major sections: 1. Logic Programming, in which we give a basic introduction to the syntax and semantics of logic programs and explain proof search; 2. Representation, in which we show how LP can be used to build expressive theory representations; and 3. Inference, in which we show how LP can be used for scientific inference.

1. Logic programming

A logic program can be written and run in a Prolog environment. In the current project we use SWI Prolog (Wielemaker et al., 2012), which we complement with the software package Theory Toolbox (Rohner, 2020). Theory Toolbox provides mechanisms for writing classical logic syntax[4] and a set of tools for scientific inference (which we present in section 3. Inference). SWI Prolog is available in this link[5] under the conditions of a BSD license. Theory Toolbox is available as a GitHub repository, in this link,[6] under the conditions of a GPL3 license

[3] Interestingly, though, there is a formal language that reads as natural language: Attempto Controlled English (ACE; Fuchs et al., 2008). The advantage of ACE is that it is easy to comprehend while still being computable (for logical inference). As far as we understand, however, the combination of Prolog and CLPR (which we use in this project) is more powerful than ACE when it comes to representing and reasoning about mathematical equations and relations that involve complex nesting patterns (both of which are important in psychology, as we explain in the paper). Still, ACE is an attractive option if having a formal representation that reads as natural language is imperative. More information about ACE can be found in this link: http://attempto.ifi.uzh.ch/site/resources/.

[4] For readers that are familiar with Prolog, term_expansion/2 and goal_expansion/2 are used to replace :- with $\Leftarrow$, comma with $\wedge$, semicolon with $\vee$, and \+ with $\neg$.

[5] https://www.swi-prolog.org/.

[6] https://github.com/JeanChristopheRohner/theory-toolbox.

(installation and operation instructions can be found in the repository).

1.1. Syntax and semantics

Fig. 1 shows a simple logic program about positive reinforcement and drug abuse (the example is fictional). Before going into details we will try to give a bird's eye view of the program. Line 1 includes[7] the code in Theory Toolbox so that symbols from classical logic can be used in the theory program. Lines 3–9 name a set of objects and describe their properties, e.g. that h1 is a human, that heroin is an opiate and that LSD is a hallucinogen. Lines 11 and 12 state two relations between objects: that h1 used heroin with a probability of 0.90 and that h2 used LSD with a probability of 0.90. Lines 14 and 15 are conditional statements; respectively, they mean that if a substance is an opiate then it causes pleasure with a probability of 0.90, and if a substance is a hallucinogen then it causes pleasure with a probability of 0.50. Lines 17–21 mean that if somebody used a substance with probability $X1$, and if that substance causes pleasure with probability $X2$, then he or she uses that substance with probability $X3 = X1{*}X2$ (the product means conjunction).

The basic components of a logic program, like the one in Fig. 1, are atomic formulas (or "atoms" for short).[8] An atomic formula represents a property or a relation that involves singular objects or sets of objects. It consists of a predicate name written with an initial lower case letter directly followed by a comma delimited list of arguments in parentheses; e.g. `opiate(heroin)` or `event(H, uses, S, X3)`. Each argument in an atomic formula is a term. A term is either a constant, a variable or a function. A constant denotes a singular object, and it is written as a string that starts with an initial lower case letter or as a number (e.g. `h1`, `heroin`, `0.90`). A variable is a placeholder that can be instantiated with, i.e. bound to, any kind of term; it is written as a string that starts with an uppercase letter (e.g. `X`, `H`, `Behavior`). A function denotes a relation and it consists of a function name directly followed by any number of terms in parentheses; in the atom `event(H, represent, event(H, like, S))`, the part `event(H, like, S)` is a function for example. Note that the names we used to denote objects, variables and relations in Fig. 1 are arbitrary. So instead of `h1`, `S` and `event(H, uses, S, X3)`, we could just as well have written, say, `person1`, `Drug` and `episode(P, uses, Drug, X3)`, respectively. Syntactically, the last three elements are still a constant, a variable and an atomic formula (in this order).

In addition to atomic formulas, a logic program can also contain numerical constraints that represent any mathematical equations in a theory; e.g. about how the probabilities of certain events are related.[9] A numerical constraint is written in curly braces, e.g. `{X1 = 1 - X2 *X3}` or `{X1 > abs(X2 - X3)}`. This functionality is provided by the CLPR module in SWI Prolog (Holzbaur, 1995), which is loaded by Theory Toolbox. A detailed specification of legal CLPR expressions, as well as the limitations of this module, can be found in Holzbaur (1995).

A logic program consists of a set of definite clauses, each delimited with a period. A definite clause is an implication with a single non-negated atomic formula in the consequent, and zero or more atomic formulas or numerical constraints in the antecedent[10]. Let C denote a non-negated atomic formula, and let A denote an atomic formula or a numerical constraint. Then the definite clause C. means that C is unconditionally provable (because the antecedent is empty); the definite clause $C \Leftarrow A_1 \wedge A_2 \wedge \ldots \wedge A_n$. means that C is provable if $A_1 \wedge A_2 \wedge \ldots \wedge A_n$ is provable. Consider the program in Fig. 1. According to the definition, the consequents on lines 3–12 are provable without further conditions; the consequents on lines 14–17 are provable if their antecedents are provable.

In the antecedent of a clause, atoms and constraints can be combined with conjunction ($\wedge$), disjunction ($\vee$), equality ($=$), or ordinary parentheses. A conjunction $A_1 \wedge A_2 \wedge \ldots \wedge A_n$ means that each A_i ($i = 1, 2, \ldots, n$.) has to be provable. A disjunction $A_1 \vee A_2 \vee \ldots \vee A_n$ means that at least one A_i ($i = 1, 2, \ldots, n$.) has to be provable. Equality can be used to state that two constants have to be equal (equality stands for unification; we return to what this means later). Parentheses are used to indicate precedence, as usual. Also, any atomic formula A_i in the antecedent of a clause can be negated by prefixing it with the symbol $\neg$, meaning that A_i is not provable. Note that $\neg A_i$ is distinct from stating that A_i is false in the sense that A_i is not the case. That A_i is not the case has to be indicated explicitly, e.g. in terms of a probability of 0: $p(A_i, 0)$, meaning that it is provable that A_i has a probability of 0.

Clauses that have a non-empty antecedent typically contain variables that denote sets of objects. Variables play an important role for expressing the idea that one or more relations hold for *all* objects of a certain kind; i.e. to make universal statements (see lines 14–21 in Fig. 1 for example). The scope of a variable is the clause in which it appears. This means that same-named variables that appear in the same clause (i.e. everything up to the period) have to be instantiated with the same term. Same-named variables that appear in different clauses can be instantiated with different terms. So in the clause that starts on line 17 in Fig. 1, the variable `S` has to be instantiated with the same constant throughout the clause (e.g. `heroin`).

1.2. Proof search

Given a logic program, such as the one in Fig. 1, search for a proof can be initiated by posing a query goal, e.g. `?-event(H, uses, S, X3)`. This search finds any variable instantiations in the goal that are provable from the program, e.g. `H = h1`, `S = heroin`, and `X3 = 0.81` from the toy example. How does this process work?

One of the key ingredients is unification (indicated by =). Unification determines if two expressions match or not according to the following rules:

1. Two constants unify if they are the same; e.g. `lsd = lsd` and `1 = 1`.
2. A variable unifies with any kind of term and is instantiated to that term; e.g. `S = heroin`, `X2 = 0.50`, `X = Y`, and `Z = event(h1, used, heroin, 0.90)`.
3. Two atomic formulas, or functions, unify if all these conditions hold:
   a. they have the same name
   b. they have the same number of arguments
   c. all of their arguments unify
   d. their variables can be instantiated consistently

Accordingly, `human(X) = human(somebody)`, because both atoms have the same name, the same number of arguments, and because a variable unifies with any kind of term (`X` unifies with `somebody`). Also `event(h, represents, Representation) = event(h, represents, event(H, likes, S))`, because both atoms have the same name, the same number of arguments, and because a variable (here `Representation`) unifies with any kind of term, including a function (here `event(H, likes, S)`). However, `something(X, X)` and `something(1, 2)` do not unify, because the instantiations of `X` are inconsistent (`X` can't be instantiated to both `1` and `2` in the same clause).

[7] Another way to achieve the same functionality is to use a Prolog module (we will try to implement this in the future when we have more time).

[8] Note that we adopt the terminology from classical first order logic as in other texts about logic programming (e.g. Nilsson & Maluszynski, 1995; Riguzzi, 2018). Some Prolog texts have a different terminology: For example, an atomic formula is sometimes called a "compound term", and a textual constant an "atom".

[9] Such constraints are not limited to probability equations, though: Expressions involving any real numbers are allowed.

[10] In technical terms a definite clause with an empty antecedent is called a "fact", and a definite clause with a non-empty antecedent is called a "rule".

```
1   :-include('theoryToolbox.pl').
2
3   human(h1).
4   human(h2).
5
6   opiate(heroin).
7   opiate(oxycodone).
8   hallucinogen(lsd).
9   hallucinogen(mescaline).
10
11  event(h1, used, heroin, 0.90).
12  event(h2, used, lsd, 0.90).
13
14  cause(S, pleasure, 0.90) ⇐ opiate(S).
15  cause(S, pleasure, 0.50) ⇐ hallucinogen(S).
16
17  event(H, uses, S, X3) ⇐
18      human(H)
19      ∧ event(H, used, S, X1)
20      ∧ cause(S, pleasure, X2)
21      ∧ {X3 = X1 * X2}.
```

Fig. 1. Toy example.

In the search for a proof a goal is provable in each one of these three important cases (without going into too much technical detail):

1. The goal is true, as in `{2 < 3}` or `{0.90 * 0.90 = 0.81}`.
2. The goal unifies with the consequent of a clause that has an empty antecedent, like `event(h2, used, lsd, 0.90).` on line 12 in Fig. 1.
3. The goal unifies with the consequent of a clause that has a provable antecedent: In $C \Leftarrow A_1 \wedge A_2 \wedge \ldots \wedge A_n$ the consequent C is provable if each A_i $(i = 1, 2, \ldots, n)$ is provable. Proving any A_i might, in turn, involve proving other antecedents (e.g. proving the antecedent of a clause in which the atom A_1 is the consequent, and so on).

Consider the toy example in Fig. 1 again and suppose that the query goal is `?-event(H, uses, S, X3)`. What is the process that leads up to a conclusion? This goal unifies with the consequent on line 17; so the next steps are as follows with regard to the antecedents of this clause:

1 `human(H)` unifies with the consequent on line 3; `H = h1`
2 `event(h1, used, S, X1)` unifies with the consequent on line 11; `S = heroin`, `X1 = 0.90`
3 `cause(heroin, pleasure, X2)` unifies with the consequent on line 14, and `opiate(heroin)` unifies with the consequent on line 6; `X2 = 0.90`
4 `{0.81 = 0.90 * 0.90}` is true

So one of the results of this search is `event(h1, uses, heroin, 0.81)`, meaning that h1 uses heroin with a probability of 0.81.

Before ending our section on LP we have to point out two important characteristics of proof search. First, how does the conclusion that a consequent is provable, in the sense we used above, relate to whether the consequent is true in the sense of corresponding to states of affairs? The answer is that it depends. Prolog uses resolution to find a proof (Robinson, 1965; Russell et al., 2010). Resolution produces valid conclusions, i.e. conclusions that have to be true *if* the premises are true. So *if* the clauses in a program are true with respect to states of affairs, then the conclusions that follow from the program are true in the same sense. In contrast, if the clauses in the program are false with respect to states of affairs, then the conclusions that follow from the program are false, or true by accident (with respect to states of affairs). The only thing that resolution provides is an answer to what a program entails. Secondly, what does it mean, exactly, that a consequent is not provable? This simply means that the consequent is not entailed by the program. So a program that misses to represent certain facts in some knowledge domain, will fail to derive any conclusions that involve these facts (the program in Fig. 1, for example, does not say anything about cocaine being reinforcing). Also note that C not being provable is distinct from deducing that C is false in the sense of not being the case (similarly to the difference between $\neg A_i$ and $p(A_i, 0)$ discussed previously). For example, the clause `animal(A) ⇐ breathes(A).` does not entail that a rock is not an animal. To be able to deduce this, we would instead have to write something like `p(animal(A), X1) ⇐ p(breathes(A), X2) ∧ {X1 = 1 * X2}.` and `p(breathes(rock), 0).`

With the technical details of LP out of the way, we can move on to a discussion of its advantages with respect to theory representation and scientific inference. We start with representation.

2. Representation

In this section we describe how psychological theories can be represented in logic programs and discuss the advantages of such representations. We start with a description of some general design principles, then give 5 theory examples, and finish with a discussion of the positive features of LP with regard to theory representation.

2.1. Design principles

A theory program is a set of definite clauses that contain a qualitative and quantitative description of events, relations between events, together with any background assumptions on which these relations depend. As stated in section 1. Logic Programming a definite clause is an implication where the consequent is provable if its antecedent is provable or, alternatively, where the consequent is unconditionally provable (if its antecedent is empty). The qualitative part of a theory consists of a description of the semantic content of each relation and its events. The quantitative part consists of set of equations that describe how probabilities of events are related. For practical purposes we divide the clauses in a theory into main clauses and background clauses. Main clauses describe the core relations of a theory; e.g. how each emotion depends on a set of antecedent conditions in a theory of emotion. Background clauses describe things that are taken for granted in the theory, like for what sets of objects the theory holds and any temporal constraints on the relations referenced in the theory (e.g. that the emotion theory holds for human beings, and that an emotion eliciting event precedes the experience of emotion).

The first principle is that the consequents that appear in the main clauses of a psychological theory describe continuous probabilities of events instead of a binary true/false outcomes (without committing to a frequentist or Bayesian interpretation). The probability of a consequent event, in turn, depends on one or more probabilistic conditions that are specified in the antecedent of the respective clause. For example, that the probability of using a drug in the future is a function of the probability of using the drug in the past and the probability that the drug caused pleasure. The probability equations are the well-known Boolean ones; i.e. $X1 {*} X2$ for conjunction, $X1 + X2 - X1 {*} X2$ for disjunction, and $1 - X$ for negation (see for example Jaynes, 2003). By having predicates that have a probability value as one of the arguments it is possible to represent degrees of truth, including falsity. Accordingly, we take 1 to mean true, i.e. that something *is* the case, and 0 to mean false, i.e. that something is not the case. So, our previous example `p(breathes (rock), 0)` means that it is not the case that a rock breathes. As stated before, stating that a consequent is false in this sense, is distinct from the situation in which the consequent is not provable.

The second principle is that we model causal relations as implications in which the effect is the consequent, and in which any number of causal factors appear as atoms in the antecedent.[11] As Mackie (1974), we believe that events typically have a plurality of causes. Any given event can occur because of several distinct clusters of factors, where multiple factors *within* a single cluster are jointly necessary and sufficient for the event to occur, while no individual cluster, in itself, is necessary for the event to occur. That is, there might be other clusters of factors that can make the same event happen (for an example see this footnote[12]). So explaining something consists in providing one or more many-to-one relations. In clausal form logic an explanatory cluster thus becomes this relation between formulas: $C \Leftarrow A_1 \wedge A_2 \wedge \ldots \wedge A_n$ (where n is an arbitrary integer); that is, the event C causally depends on the set of necessary and sufficient factors $A_1, A_2, \ldots, A_n$. And the same event C, might occur because of distinct clusters of factors, e.g. $C \Leftarrow A_1 \wedge A_2 \wedge A_3$, or $C \Leftarrow A_4 \wedge A_5 \wedge A_6 \wedge A_7$ and so on. With that said, it is important to remember that implication relations are distinct from causal relations. Causal relations make much stronger ontological commitments, e.g. about the temporal order between causes and effects, and about what happens with an effect when one or more of its causes are manipulated (e.g. Pearl, 2009; Shadish et al., 2002). The only meaning of an implication relation is that the consequent follows deductively from the antecedent. It is up to whoever designs a theory to specify what clauses state causal relations and what clauses state other kinds of relations.

The third design principle is that we decompose construct names into their semantic components (a construct is a concept that stands for a psychologically relevant attribute or process, e.g. intelligence, exercise behavior, or memory). Note that we used the word "event" instead of "construct" or "variable" when describing relata in the clauses of a theory. To this end we assume that construct names actually reference more or less complex psychological events and that these events typically have 5 basic components, roughly following Pylyshyn (1973): A subject, a verb, an object, a time and a value.

1. The subject is the thing that does something or is something.
2. The verb is what is being done.
3. The object is the thing that is acted upon by the subject.
4. Time is a more or less extended time frame.
5. Value is a number that indicates the probability of the event.

For example, to encode the meaning of the set of events that the construct name "exercise behavior" references, our approach is to write `event(H, perform, exercise, T, X)` $\Leftarrow$ `human(H)` $\wedge$ `time(T)`, which means that any human performs exercise with probability X at any time. We also assume that the subject and object arguments in the event predicate can be other events. Sometimes, for example, the object of a person's thoughts is another event. This statement says that the probability that somebody thinks that it is unlikely that smoking causes cancer in the future is 0.5: `event(H, represent, event(smoking, cause, cancer, T1, 0), T2, 0.5)` $\Leftarrow$ `human(H)` $\wedge$ `precedes(T2, T1)`. Note that this sentence distinguishes between the semantic content and probability of the act of representing something (in the outer event), and the semantic content and probability *in* the representation (in the inner event, which describes the object of the act of representing).

2.2. Theory examples

In this section we present 5 examples that illustrate how psychological theories can be represented in logic programs. By having a variety of examples we intend to illustrate that LP can handle a number of different scenarios. Note that all examples are desktop products which haven't been pitted against empirical observations. Our goal is to illustrate how LP can be used for theory representation, not to propose new theories per se (we tried to mirror the respective theory as closely as possible though). Each theory program starts with the directive`:-include('theoryToolbox.pl').` and a part called "INPUT". We will explain what these statements mean in more detail in section 3. Inference; for now we just skip them (the `%` symbol denotes a comment, i.e. text that isn't code).

2.2.1. Phobia

Fig. 2 shows a basic theory program that represents the important parts of definition of simple phobia from the Diagnostic and Statistical Manual of Mental Disorders (DSM5; American Psychiatric Association, 2013). The background clauses of the theory define a set of objects of phobia (derived from the examples section in the DSM5). Main clauses 1, 2, and 3 list different outcomes of the disorder; main clause 4 defines an assumption that is external to the DSM5 definition. Consider, for example, what main clause 1 means: The probability that H fears O is $X1$ if the following conditions hold: Source is the DSM5, H is a human, O is an object, the probability that H has a phobia for O is $X2$, the probability of $X1$ is equal to $X2$. The other clauses in the theory can be interpreted in a similar way.

Because this is our first theory example, we take the opportunity to point out some important features of the program (please look out for these features in the other examples as well). First, the antecedents of each main clause contain a set of domain definitions: The `H`s stand for humans and the `O`s stand for objects (the use of braces places the `X`s in the domain of real numbers, per the CLPR module). Describing the domain of each variable in the clauses of a theory is good practice, because this explicitly declares the sets of objects that are within the scope of the theory (for example, given that the `H`s are declared to be humans, the theory explicitly says that it does not hold for other beings). Secondly, there is an atom named `source` in the main clauses of the theory. This is a way to keep track of the clauses that are involved in a certain proof, and to study what happens when a certain clause is assumed to hold, or assumed not to hold (we have more to say about this in section 3. Inference). Third, note that from this theory it is possible to deduce that some things are *not the case*, as opposed to being not provable. For example, if the probability of phobia is zero, then the probability of fear will be zero.

Even if the phobia example in Fig. 2 has advantages from a didactic

[11] Readers who are familiar with logic programming might wonder why we do not use a predicate to represent causal relations, schematically *causes*($[P_1, P_2, \ldots, P_n], E$), where the Ps and E are atoms that represent the preconditions for an effect and the effect, respectively (a list is necessary because the number of preconditions for something usually varies). There are two reasons for why we avoid this pattern. First, we think that it is harder to understand. Whenever any of the Ps or the E contain variables, that represent probabilities or other domains, these have to be bound in an antecedent. The resultant clauses have the pattern *causes*($[P_1, P_2, \ldots, P_n], E$)$\Leftarrow A_1, A_2, \ldots, A_m$ (where the As stand for atoms or constraints in the antecedent). Assuming that people (informally) picture both implication relations and causal relations as "if-then" relations, the mental image ends up being "If $A_1, A_2, \ldots, A_m$, then (if $P_1, P_2, \ldots, P_n$, then E)", which is convoluted. Secondly, we have noted that there is a risk for ambiguity: *Something* needs to be placed in the antecedent to bind any variables in the consequent, but what? Suppose for example that E stands for fire; should flammability, presence of oxygen or presence of heat be moved to the antecedent? There is no clear cut answer. All three conditions can't be moved to the antecedent because this eliminates the reason for having a binary causes predicate in the first place.

[12] Consider causal explanations of a behavior. One possible explanatory cluster is that a person performed the behavior in the past and that it was followed by a positive outcome. Another explanatory cluster is that the person was influenced by peers which he or she admired, and that the environment afforded the behavior. And so on.

```
:-include('theoryToolbox.pl').


% INPUT

% source(S)
% human(H)
% event(H, phobia, _, time, X)

% S = dsm5 ∨ S = plausibleAssumption
% H is a constant
% {X ∈ ℝ | 0 =< X =< 1}


% THEORY

% BACKGROUND CLAUSES

object(spiders).
object(insects).
object(dogs).
object(heights).
object(storms).
object(water).
object(needles).
object(medicalProcedures).
object(airplanes).
object(elevators).
object(enclosedSpaces).
```

```
% MAIN CLAUSES

% 1 "Marked fear or anxiety about a specific object or situation (e.g., flying,
% heights, animals, receiving an injection, seeing blood)." and "The fear, anxiety,
% or avoidance causes clinically significant distress or impairment in social,
% occupational, or other important areas of functioning."
event(H, fear, O, time, X1) ⇐
    source(dsm5)
    ∧ human(H)
    ∧ object(O)
    ∧ event(H, phobia, O, time, X2)
    ∧ {X1 = X2}.

% 2 "The phobic object or situation almost always provokes immediate fear or anxiety."
event(H, fear, O, time, X1) ⇐
    source(dsm5)
    ∧ human(H)
    ∧ object(O)
    ∧ event(H, phobia, O, time, X2)
    ∧ event(H, encounter, O, time, X3)
    ∧ {X1 = X2 * X3}.

% 3 "The phobic object or situation is avoided or endured with intense fear
% or anxiety."
event(H, avoid, O, time, X1) ⇐
    source(dsm5)
    ∧ human(H)
    ∧ object(O)
    ∧ event(H, phobia, O, time, X2)
    ∧ {X1 = X2}.

% 4 Plausible assumption
event(H, encounter, O, time, X1) ⇐
    source(plausibleAssumption)
    ∧ human(H)
    ∧ object(O)
    ∧ event(H, avoid, O, time, X2)
    ∧ {X1 = 1 - X2}.
```

Fig. 2. Phobia example.

perspective, because it is simple, it does not represent a good theory. Since it is supposed to mirror the DSM5 definition as closely as possible, it ends up having two clauses with the same consequent: main clauses 1 and 2. In conjunction with what main clause 4 entails, this leads to a problem in the theory. Can you spot it? If not, it will be revealed in section 3. Inference.

2.2.2. *Cognitive appraisal and emotion*

Fig. 3 shows an appraisal theory of emotion, which is based on the work of Lazarus (1991), Smith and Ellsworth (1985), and Smith and Lazarus (1993). The background clauses of this theory define some time frames, their relations, some goals, and how much people in general value each goal. The main clauses of the theory state how the probabilities of experiencing anger, fear, shame, sadness and happiness (`X1`s) depend on the probabilities of distinct appraisal patterns (`X2`s through `X7`s). Different appraisal patterns, in conjunction, trigger different emotions (the products in the equations represent conjunctions). An event can be appraised as congruent with a valued goal, in which case happiness occurs, or appraised as not congruent with a valued goal, in which case a negative emotion occurs (note that the probability of experiencing negative emotions is negatively related to experiencing an event as goal as congruent). Which negative emotion occurs is determined by how responsibility is attributed; i.e. blaming oneself in shame, blaming others in anger, and blaming the world in sadness.

Note that appraisal and emotional experience have different time frames than the events that trigger emotion (in the inner event). So for example, the experience of sadness occurs in relation to something that happened in a preceding time frame; the experience of fear occurs in relation to something that might happen in a future time frame.

2.2.3. *Substance misuse*

Fig. 4A and Fig. 4B shows a theory about the relations between substance misuse, physical harm, cognitive processes, operant learning, and vicarious (observational) learning. It is a larger theory that integrates four research traditions: The Theory of Planned Behavior (Ajzen, 1991), operant learning theory (Skinner, 1953), observational learning theory (Bandura, 2004), and research about the addictiveness and harmfulness of different drugs (Nutt et al., 2007). Given that addiction seems to be a complex and multifaceted phenomenon we found it interesting to show how a logic program can unify various explanations in the same formal representation.

The background clauses of this theory (shown in Fig. 4A) define what behaviors count as substance misuse, the potential outcomes of misuse, the amount of physical harm and pleasure associated with using different substances, and the time frames of the theory (1 to 6 is arbitrary; harm and pleasure values were obtained by dividing the expert ratings in Nutt et al., 2007 by 3 to get a 0 to 1 range).

The main clauses of the theory (Fig. 4B) then state the following relations: The influence of cognitive beliefs on attitudes, perceived control and perceived norms (main clauses 1, 2, and 3); the influence of attitudes, perceived control and perceived norms on behavioral intentions (4); the influence of behavioral intentions on behavior (5); the influence of operant learning on behavior (6); the influence of vicarious learning on behavior (7); and the influence of substance misuse on physical harm (8). To take an example, consider what main clause 1 means in simpler terms: Somebody represents that they like a misuse behavior with probability $X1$, if they represent that the behavior causes a positive outcome with probability $X2$, and if they represent that the behavior causes a negative outcome with probability $X3$, and if $X1 = X2 * (1 - X3)$, where the product means conjunction and where $1 - X3$ means negation of $X3$.

Note that main clauses 1, 2 and 3 contain atoms denoted `exogenousEvent` in the antecedent. These atoms do not appear as consequents in any clause of the theory, so they are not provable without additional information. As will be shown in section 3. Inference, `exogenousEvent` atoms have to be provided by the user instead. Loosely speaking, these can be thought of as theory “parameters” that can be varied by the user to explore different conclusions.

There are three characteristics of this theory which are worth discussing. First, it contains a longer chain of events than the phobia and emotion theories we showed before. The effects of cognitive beliefs on behavior, for example, are mediated by a number of events. Secondly, this theory contains a recursive clause (main clause 6). A recursive clause is a clause in which the consequent also appears as an atomic formula in the antecedent. Main clause 6 means that misuse at time $T1$ depends on misuse at time $T2$ and on whether misuse causes a positive outcome, provided that $T2$ precedes $T1$. Given the background assumptions about time frames and their relations, the theory therefore entails that misuse at time 6 is provable from misuse at time 1. Third, note that there is a weakness in this theory. Consider the clause about physical harm (8); it simply says that harm at time $T1$ depends on how harmful misuse M is and on whether somebody has performed misuse M at time $T2$. There is nothing in this clause that defines how physical harm in one time frame also depends on harm in previous time frames, but prolonged substance misuse should presumably have a cumulative effect on harm. We have made this simplification for didactic reasons, but it is possible to model a scenario in which harm is propagated across time frames (see the file substanceMisuseExampleState.pl in the GitHub repository).

2.2.4. *Transitivity of distance relations*

Fig. 5 shows a theory about how people deduce transitive distance relations. Main clause 1 simply means that people can deduce that A is beyond B if they represent that A is beyond B; and because this should be easy, the two events are stipulated to have equal probabilities. Main clause 2 is a recursive one; it means that people can deduce that A is beyond C if they represent that A is beyond B and if they can deduce that B is beyond C. Because this should be harder, the probability of the A is beyond C deduction is weighted with 0.8 (which is just a simple guesstimate).

A distinctive feature of this theory is that it can model inferences about distance relations that involve an arbitrary number of steps by just using the same two clauses (1 and 2). Suppose that this theory is handed the information *relation*(*jupiter*, *beyond*, *mars*), *relation*(*mars*, *beyond*, *earth*), and *relation*(*earth*, *beyond*, *venus*), where we have omitted the prefix *event*(*somebody*, *represent*, ...) for the sake of clarity. According to the theory, somebody should then be able to deduce *relation*(*jupiter*, *beyond*, *venus*) with probability 0.64. And adding the information *relation*(*saturn*, *beyond*, *jupiter*) would result in the conclusion that somebody can deduce *relation*(*saturn*, *beyond*, *venus*) with probability 0.51.

2.2.5. *Planning*

Fig. 6 shows a theory about how people deduce plans from their mental representations of the actions that bring about specific state transitions (e.g. the actions needed to get from the state being at work, to

```
:-include('theoryToolbox.pl').


% INPUT

% human(H1)
% human(H2)
% event(E)
% event(H1, appraise, _, present, X)

% H1, H2 and E are constants
% {X ∈ ℝ | 0 =< X =< 1}


% THEORY

% BACKGROUND CLAUSES

time(past).
time(present).
time(future).
precedes(past, present).
precedes(present, future).

goal(socialApproval).
goal(achievement).
goal(autonomy).

event(H, value, socialApproval, T, 0.7) ⇐ human(H) ∧ time(T).
event(H, value, achievement, T, 0.6) ⇐ human(H) ∧ time(T).
event(H, value, autonomy, T, 0.5) ⇐ human(H) ∧ time(T).


% MAIN CLAUSES

event(H1, experience, anger, T1, X1) ⇐
    human(H1)
    ∧ human(H2)
    ∧ ¬(H1 = H2)
    ∧ event(E)
    ∧ goal(G)
    ∧ precedes(T2, T1)
    ∧ event(H1, value, G, T1, X2)
    ∧ event(H1, appraise, event(H1, experience, E, T2, 1), T1, X3)
    ∧ event(H1, appraise, event(E, congruent, G, T2, 1), T1, X4)
    ∧ event(H1, appraise, event(H1, cause, E, T2, 1), T1, X5)
    ∧ event(H1, appraise, event(H2, cause, E, T2, 1), T1, X6)
    ∧ event(H1, appraise, event(world, cause, E, T2, 1), T1, X7)
    ∧ {X1 = X2 * X3 * (1 - X4) * (1 - X5) * X6 * (1 - X7)}.

event(H1, experience, shame, T1, X1) ⇐
    human(H1)
    ∧ human(H2)
    ∧ ¬(H1 = H2)
    ∧ event(E)
    ∧ goal(G)
    ∧ precedes(T2, T1)
    ∧ event(H1, value, G, T1, X2)
    ∧ event(H1, appraise, event(H1, experience, E, T2, 1), T1, X3)
    ∧ event(H1, appraise, event(E, congruent, G, T2, 1), T1, X4)
    ∧ event(H1, appraise, event(H1, cause, E, T2, 1), T1, X5)
    ∧ event(H1, appraise, event(H2, cause, E, T2, 1), T1, X6)
    ∧ event(H1, appraise, event(world, cause, E, T2, 1), T1, X7)
    ∧ {X1 = X2 * X3 * (1 - X4) * X5 * (1 - X6) * (1 - X7)}.

event(H, experience, fear, T1, X1) ⇐
    human(H)
    ∧ event(E)
    ∧ goal(G)
    ∧ precedes(T1, T2)
    ∧ event(H, value, G, T1, X2)
    ∧ event(H, appraise, event(H, experience, E, T2, 1), T1, X3)
    ∧ event(H, appraise, event(E, congruent, G, T2, 1), T1, X4)
    ∧ {X1 = X2 * X3 * (1 - X4)}.

event(H1, experience, sadness, T1, X1) ⇐
    human(H1)
    ∧ human(H2)
    ∧ ¬(H1 = H2)
    ∧ event(E)
    ∧ goal(G)
    ∧ precedes(T2, T1)
    ∧ event(H1, value, G, T1, X2)
    ∧ event(H1, appraise, event(H1, experience, E, T2, 1), T1, X3)
    ∧ event(H1, appraise, event(E, congruent, G, T2, 1), T1, X4)
    ∧ event(H1, appraise, event(H1, cause, E, T2, 1), T1, X5)
    ∧ event(H1, appraise, event(H2, cause, E, T2, 1), T1, X6)
    ∧ event(H1, appraise, event(world, cause, E, T2, 1), T1, X7)
    ∧ {X1 = X2 * X3 * (1 - X4) * (1 - X5) * (1 - X6) * X7}.

event(H, experience, happiness, T1, X1) ⇐
    human(H)
    ∧ event(E)
    ∧ goal(G)
    ∧ precedes(T2, T1)
    ∧ event(H, value, G, T1, X2)
    ∧ event(H, appraise, event(H, experience, E, T2, 1), T1, X3)
    ∧ event(H, appraise, event(E, congruent, G, T2, 1), T1, X4)
    ∧ {X1 = X2 * X3 * X4}.
```

Fig. 3. Emotion example.

```
:-include('theoryToolbox.pl').


% INPUT

% source(S)
% human(H)
% referent(R, H)
% exogenousEvent(_, _, _, _, X)

% S = theoryOfPlannedBehavior ∨ S = operantLearning ∨ S = vicariousLearning ∨
% S = harmBehavior
% H and R are constants
% {X ∈ ℝ | 0 =< X =< 1}


% THEORY

% BACKGROUND CLAUSES

misuse(useHeroin).
misuse(useCocaine).
misuse(useAlchohol).
misuse(useBenzodiazepines).
misuse(useAmphetamine).
misuse(useTobacco).
misuse(useCannabis).
misuse(useLSD).
misuse(useEcstasy).
misuse(useStreetMethadone).

outcome(pleasure).
outcome(physicalHarm).
positive(pleasure).
negative(physicalHarm).
reinforcer(pleasure).

time(1).
time(2).
time(3).
time(4).
time(5).
time(6).

precedes(X, Y) ⇐ time(X) ∧ time(Y) ∧ {Y = X + 1}.
```

```
event(useHeroin, cause, physicalHarm, T, 0.93) ⇐ time(T).
event(useCocaine, cause, physicalHarm, T, 0.78) ⇐ time(T).
event(useAlchohol, cause, physicalHarm, T, 0.47) ⇐ time(T).
event(useBenzodiazepines, cause, physicalHarm, T, 0.54) ⇐ time(T).
event(useAmphetamine, cause, physicalHarm, T, 0.60) ⇐ time(T).
event(useTobacco, cause, physicalHarm, T, 0.41) ⇐ time(T).
event(useCannabis, cause, physicalHarm, T, 0.33) ⇐ time(T).
event(useLSD, cause, physicalHarm, T, 0.38) ⇐ time(T).
event(useEcstasy, cause, physicalHarm, T, 0.35) ⇐ time(T).
event(useStreetMethadone, cause, physicalHarm, T, 0.62) ⇐ time(T).

event(useHeroin, cause, pleasure, T, 1.00) ⇐ time(T).
event(useCocaine, cause, pleasure, T, 1.00) ⇐ time(T).
event(useAlchohol, cause, pleasure, T, 0.77) ⇐ time(T).
event(useBenzodiazepines, cause, pleasure, T, 0.57) ⇐ time(T).
event(useAmphetamine, cause, pleasure, T, 0.67) ⇐ time(T).
event(useTobacco, cause, pleasure, T, 0.77) ⇐ time(T).
event(useCannabis, cause, pleasure, T, 0.63) ⇐ time(T).
event(useLSD, cause, pleasure, T, 0.73) ⇐ time(T).
event(useEcstasy, cause, pleasure, T, 0.50) ⇐ time(T).
event(useStreetMethadone, cause, pleasure, T, 0.60) ⇐ time(T).
```

Fig. 4A. Substance misuse example: Background clauses.

```
% MAIN CLAUSES

% 1 Theory of planned behavior: Expectancy-value beliefs and attitudes
event(H, represent, event(H, like, M, T1, 1), T1, X1) ⇐
    source(theoryOfPlannedBehavior)
    ∧ human(H)
    ∧ misuse(M)
    ∧ outcome(PO) ∧ outcome(NO)
    ∧ positive(PO) ∧ negative(NO)
    ∧ precedes(T2, T1)
    ∧ exogenousEvent(H, represent, event(M, cause, PO, T2, 1), T2, X2)
    ∧ exogenousEvent(H, represent, event(M, cause, NO, T2, 1), T2, X3)
    ∧ {X1 = X2 * (1 - X3)}.

% 2 Theory of planned behavior: Control beliefs and perceived control
event(H, represent, event(H, control, M, T1, 1), T1, X1) ⇐
    source(theoryOfPlannedBehavior)
    ∧ human(H)
    ∧ misuse(M)
    ∧ precedes(T2, T1)
    ∧ exogenousEvent(H, represent, event(environment, afford, M, T2, 1), T2, X2)
    ∧ exogenousEvent(H, represent, event(H, affect, environment, T2, 1), T2, X3)
    ∧ {X1 = X2 * X3}.

% 3 Theory of planned behavior: Norm beliefs and perceived norms
event(H, represent, event(H, should, M, T1, 1), T1, X1) ⇐
    source(theoryOfPlannedBehavior)
    ∧ human(H)
    ∧ referent(R, H)
    ∧ misuse(M)
    ∧ precedes(T2, T1)
    ∧ exogenousEvent(R, represent, event(H, should, M, T2, 1), T2, X2)
    ∧ exogenousEvent(H, represent, event(H, comply, R, T2, 1), T2, X3)
    ∧ {X1 = X2 * X3}.

% 4 Theory of planned behavior: Attitude, control, norm and intention
event(H, represent, event(H, intend, M, T1, 1), T1, X1) ⇐
    source(theoryOfPlannedBehavior)
    ∧ human(H)
    ∧ misuse(M)
    ∧ precedes(T2, T1)
    ∧ event(H, represent, event(H, like, M, T2, 1), T2, X2)
    ∧ event(H, represent, event(H, control, M, T2, 1), T2, X3)
    ∧ event(H, represent, event(H, should, M, T2, 1), T2, X4)
    ∧ {X1 = X2 * X3 * X4}.

% 5 Theory of planned behavior: Intention and behavior
event(H, perform, M, T1, X1) ⇐
    source(theoryOfPlannedBehavior)
    ∧ human(H)
    ∧ misuse(M)
    ∧ precedes(T2, T1)
    ∧ event(H, represent, event(H, intend, M, T2, 1), T2, X2)
    ∧ {X1 = X2}.

% 6 Operant learning: Positive reinforcement
event(H, perform, M, T1, X1) ⇐
    source(operantLearning)
    ∧ human(H)
    ∧ misuse(M)
    ∧ outcome(O)
    ∧ positive(O)
    ∧ reinforcer(O)
    ∧ precedes(T2, T1)
    ∧ event(M, cause, O, T2, X2)
    ∧ event(H, perform, M, T2, X3)
    ∧ {X1 = X2 * X3}.

% 7 Vicarious learning: Positive reinforcement
event(H, perform, M, T1, X1) ⇐
    source(vicariousLearning)
    ∧ human(H)
    ∧ misuse(M)
    ∧ referent(R, H)
    ∧ outcome(O)
    ∧ positive(O)
    ∧ reinforcer(O)
    ∧ precedes(T2, T1)
    ∧ event(M, cause, O, T2, X2)
    ∧ event(R, perform, M, T2, X3)
    ∧ exogenousEvent(H, attend, R, T2, X4)
    ∧ exogenousEvent(H, capable, M, T2, X5)
    ∧ {X1 = X2 * X3 * X4 * X5}.

% 8 Misuse behavior and physical harm
event(H, experience, physicalHarm, T1, X1) ⇐
    source(harmBehavior)
    ∧ human(H)
    ∧ misuse(M)
    ∧ precedes(T2, T1)
    ∧ event(H, perform, M, T2, X2)
    ∧ event(M, cause, physicalHarm, T2, X3)
    ∧ {X1 = X2 * X3}.
```

Fig. 4B. Substance misuse example: Main clauses.

```
:-include('theoryToolbox.pl').


% INPUT

% source(S)
% human(H)
% event(H, represent, relation(A, beyond, B), time, X)

% H, A and B are constants
% S = recursive ∨ S = nonrecursive
% {X ∈ ℝ | 0 =< X =< 1}


% THEORY

% MAIN CLAUSES

% 1 Base case
event(H, deduce, relation(A, beyond, B), time, X1) ⇐
    human(H)
    ∧ event(H, represent, relation(A, beyond, B), time, X2)
    ∧ {X1 = 1 * X2}.

% 2 Recursive clause
event(H, deduce, relation(A, beyond, C), time, X1) ⇐
    source(recursive)
    ∧ human(H)
    ∧ event(H, represent, relation(A, beyond, B), time, X2)
    ∧ event(H, deduce, relation(B, beyond, C), time, X3)
    ∧ {X1 = 0.8 * X2 * X3}.

% 3 Non-recursive clause
event(H, deduce, relation(A, beyond, C), time, X1) ⇐
    source(nonrecursive)
    ∧ human(H)
    ∧ event(H, represent, relation(A, beyond, B), time, X2)
    ∧ event(H, represent, relation(B, beyond, C), time, X3)
    ∧ {X1 = 0.8 * X2 * X3}.
```

Fig. 5. Distance example.

the state being at home). A plan consists of a start state and a sequence of actions that lead up to a goal state. There are two main clauses in the theory. The first says that it is possible to deduce a plan from a representation of a state transition. The second says that is possible to deduce a plan by combining a representation of a state transition and a deduced plan (which is harder, hence the 0.8 weight in the product of the last clause).

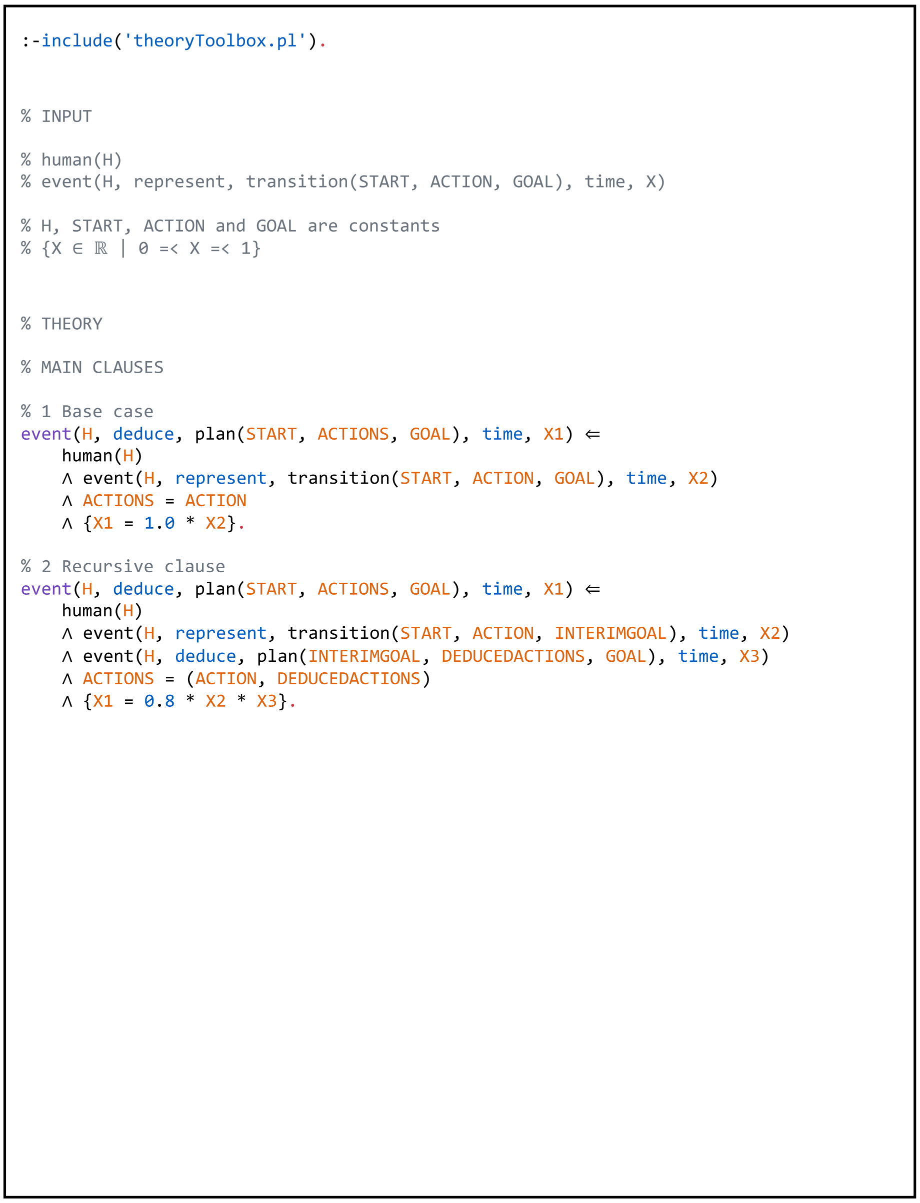
A characteristic feature of this theory is that it can model events that have a variable number of components. Cognitive representations of plans usually have one start state and one goal state, but the number of actions needed to get from start to goal varies a lot depending what kind of plan one is considering. This is achieved by using recursion, where the second clause calls itself and the first clause repeatedly, appending `DEDUCEDACTIONS` to `ACTION` each time. Suppose that this theory is handed information about the following state-action-state transitions, excluding the prefix *event*(*H*, *represent*, ...) for the sake of simplicity: *transition*(*s*1, *a*1, *s*2), *transition*(*s*2, *a*2, *s*3), and *transition*(*s*3, *a*3, *s*4). The theory would then entail *plan*(*s*1, (*a*1, (*a*2, *a*3)), *s*4). And if the input information is augmented with *transition*(*s*4, *a*4, *s*5) and *transition*(*s*5, *a*5, *s*6), for example, it would also entail *plan*(*s*1, (*a*1, (*a*2, (*a*3, (*a*4, *a*5)))), *s*6). The theory therefore represents events that have a variable number of components.

2.3. Advantages with respect to representation

In this section we highlight the advantages of LP for representing scientific theories in psychology. We discuss five positive features of LP and refer back to the theory examples in the previous section.

2.3.1. Explicit background assumptions

All theories presuppose a more or less complex web of background assumptions that determine what explanations and predictions they entail (e.g. Stanford, 2017). Such assumptions might involve restrictions on the kinds of behaviors, cognitions and populations that fall within the scope of a theory, as well as auxiliary hypotheses about the workings of relevant observational methods. Say that a hypothesis is that intentions to perform a behavior cause the behavior. In an empirical test of this hypothesis it is unavoidable to pick an instance of a behavior and a



```
:-include('theoryToolbox.pl').


% INPUT

% human(H)
% event(H, represent, transition(START, ACTION, GOAL), time, X)

% H, START, ACTION and GOAL are constants
% {X ∈ ℝ | 0 =< X =< 1}


% THEORY

% MAIN CLAUSES

% 1 Base case
event(H, deduce, plan(START, ACTIONS, GOAL), time, X1) ⇐
    human(H)
    ∧ event(H, represent, transition(START, ACTION, GOAL), time, X2)
    ∧ ACTIONS = ACTION
    ∧ {X1 = 1.0 * X2}.

% 2 Recursive clause
event(H, deduce, plan(START, ACTIONS, GOAL), time, X1) ⇐
    human(H)
    ∧ event(H, represent, transition(START, ACTION, INTERIMGOAL), time, X2)
    ∧ event(H, deduce, plan(INTERIMGOAL, DEDUCEDACTIONS, GOAL), time, X3)
    ∧ ACTIONS = (ACTION, DEDUCEDACTIONS)
    ∧ {X1 = 0.8 * X2 * X3}.
```

Fig. 6. Planning example.

sample from a population, and doing so introduces two auxiliary hypotheses that will determine whether the theory succeeds of fails. This is known as the Duhem-Quine thesis: Any singular hypothesis cannot be refuted in light of contradicting observations, because predictions that are derived from the hypothesis also depend on other auxiliary hypotheses (Bunnin & Yu, 2008). A major problem arises when such background assumptions are not stated explicitly: It is difficult to say whether a set of observations contradicts a theory or not, and proponents of the theory may opportunistically claim that the set of observations was, or was not, outside the scope of the theory. The problem is well-known in philosophy of science (e.g. Popper, 1972).

In a logic program it is possible to build a formal description of properties and relations involving any kinds of objects besides numbers. So in addition to encoding the core relations of a theory it is also possible to give a clear and detailed specification of its background assumptions. Consider the emotion theory and the substance misuse theory (Figs. 3, 4A and 4B). These theories contain a rather detailed list of background clauses, e.g. about how much people value certain goals and about what behaviors count as misuse. If some or all of these clauses are removed, important parts of the theories simply become unprovable, because all background clauses appear in the antecedents of one or more main clauses. The upshot is that architects of a theory are encouraged to be specific about their presuppositions; and when they are, their theory will generate unambiguous answers. Consider the following cases in relation to the substance misuse theory. Does it predict that intentions to use cannabis lead to cannabis use? Yes. Does it predict that intentions to use methylphenidate lead to using methylphenidate? No; `methylphenidate` does not appear in the background clauses. Does it predict cannabis use in teenagers? Yes; the domain of `H` is all human beings. And do events that are perceived to be incongruent with survival goals cause negative emotions in the emotion theory? No, such a goal doesn't exist in the background clauses (although it probably should). The point is that the use of LP involves a commitment to be explicit and precise about all the parts of a theory, qualities that are important in any scientific project.

2.3.2. *Intra event semantics*

Instead of construct names our design principles advise the use of event predicates as the basic building blocks of the relations in a formal theory; e.g. `event(H, represent, event(M, cause, PO, T2, 1),` `T2, X2)` instead of a symbol such as "behavioral beliefs" (here `H`, `M`, `PO`, `T2`, and `X2` refer to humans, misuse behaviors, positive outcomes, time frames, and probabilities, respectively). The goal is to capture the meaning of the events that construct names reference. As stated previously, the event predicate has five arguments: Subject, verb, object, time and value. Subject, verb and object match the components that are found in almost all natural language sentences. Time is a universal property of events; something always happens in some time frame, even things that at first may appear static, like properties of people. Time is an integral part of memory theory, developmental theory, learning theory, and so on, but it also plays a more general role, given that causal relations and prediction relations are asynchronous.[13]

Being able to nest event predicates is important for expressing subtle nuances in meaning. For example, it is possible to state that emotional appraisal in a certain time frame is about an event that happened or might happen in a different time frame (see sadness and anxiety in Fig. 3), or that a person's beliefs about his or her control over a behavior are distinct from that person's actual control over the behavior (see main clause 2 in Fig. 4B). Logic programming does not impose an upper limit on the number of functions that can be nested in an atomic formula.[14] So it is possible to make precise statements about complex constructs like, say, higher order theory of mind (e.g. Baron-Cohen, 2000): That person 1 thinks about what person 2 thinks about what person 3 thinks: `event(H1, represent, event(H2, represent, event(H3, represent, R, T, 1), T, 1), T, 1)` where the `H`s stand for humans, `R` for a representation and `T` for a time frame.

By using variables it is possible to formalize some important *intra* event relations. Consider the appraisal patterns for anger and shame in the emotion theory (Fig. 3). By having different combinations of `H1` and `H2` in the inner and outer event, it is possible to make detailed statements about the attribution patterns that are involved in anger and shame. And in the substance misuse theory in Fig. 4B (main clause 5), it is possible to say that the person that has an intention and the person in the contents of the intention are one and the same (by having the same variable `H` in the outer and inner event). This is not nitpicking: Sometimes a theory is supposed to say something else, e.g. about our intentions towards the behaviors of others. Consider another example which illustrates the use of variables within an event predicate. Suppose that the goal is to make formal statements about a construct such as memory for self-performed actions; according to our design principles this would be `event(H, represent, event(H, perform, A, T1, X1), T2, X2)`, with the antecedents `human(H)`, `action(A)`, `time(T1)`, `time(T2)`, `precedes(T1, T2)`, where `X1` and `X2` are numbers. By having `H` both in the inner and outer event, `T1` and `T2` in the inner and outer event (respectively), and the aforementioned constraints, it is possible to make exact statements about the meaning of this construct.

Another advantage with having event predicates instead of construct names as building blocks, is that a limited set of symbols can be used to express several different concepts in a parsimonious way. This is because logic is a compositional language. Consider the construct names in Table 1, which all appear in the scientific literature. A closer look reveals that all constructs involve the same objects (persons, groups and probabilities) and the same relations (liking and belonging) in different combinations. The corresponding LP statements are shown in the second column of Table 1 (where the last argument in the event predicate is a probability and where we have omitted the time argument for the sake of simplicity).

Compare this to using construct names as the building blocks of formal theories. If construct names reference relations, as in Table 1, and if there are *N* objects that can take part in such relations, the number of construct names becomes as large as all the ways in which these *N* objects can be (meaningfully) combined into relations. Short names for important scientific concepts are practical for quick and easy communication. But a formal language in which the number of names might increase exponentially as a function of the number of things one wants to say is impractical.

The last advantage of LP with respect to describing the meaning of events, is its capacity to represent events that have a variable number of components. Consider the planning example in Fig. 6 again. In the consequent `event(H, deduce, plan(START, ACTIONS, GOAL), time, X1)`, the variable `ACTIONS` can hold any number of actions, because each recursion in the second clause appends one or more deduced actions to an action. Planning is just one application of recursion, but to us it seems that this property of LP could be important for modeling several different kinds of cognitive processes. To name a few: Reasoning about causal relations, spatial cognition, development of vocabulary size, and so on. Essentially, LP should come in handy whenever one wants to model mental representations that contain a variable number of objects.

2.3.3. *Inter event semantics*

Besides being able to formalize construct meaning by using the event predicate, our design principles also enable exact statements about the *intra-event* components of *different events* in a clause. Schematically, suppose that a clause contains an antecedent $event(S1, V1, O1, T1, X1)$ and a consequent $event(S2, V2, O2, T2, X2)$, where the arguments refer to subject, verb, object, time and value as usual. In this scheme, it is possible to make statements about any relevant relations between $S1$ and $S2$, $V1$ and $V2$, $O1$ and $O2$, and $T1$ and $T2$, as well as any relations that mix these arguments, such as $S1$ and $V1$, $P1$ and $T2$, and so on. This feature plays an important role in many situations. Consider how the emotions of shame and anger are described in Fig. 3 (main clauses 1 and 2), for example. By having `H1` and `H2` in different events across the clause and the restriction $\neg$`(H1 = H2)` it is possible to state the following exactly: A person experiences shame if that person thinks that he or she caused an event that he or she perceives to be incongruent with a goal that he or she values; a person experiences anger if that person experiences an event that he or she appraises to be incongruent with a valued goal, and if he or she thinks that another person caused the event. And in the substance misuse example (Fig. 4A and Fig. 4B), the use of the variable `M` across the events in main clause 4 explicitly conveys the following information: The object of a person's intention and the object in that person's attitudinal-, control- and normative beliefs, are exactly one and the same. This is not a trivial statement: A well-known problem in attitude research is that behaviors at different levels of abstraction have been confounded, e.g. attitudes towards religion and actually going to church (Fishbein & Ajzen, 1974). Finally, consider an example clause that describes recognition memory and attention for any stimulus `S`: `event(H, represent, S, T1, X1)` $\Leftarrow$ `event(world, expose, S, T2, X2)` $\wedge$ `event(H, attend, S, T2, X3)`. It goes without saying that the relations between subjects, verbs, objects, time frames and probabilities in these three formulas are essential in the hypothesis.

Another powerful feature of LP with regard to capturing the inter event relations of a theory is recursion, as mentioned earlier. With recursion it is possible to state general relations between events in a compact form. Consider the substance misuse example in Fig. 4A and Fig. 4B. Here, main clause 6 (which is recursive) models positive reinforcement across *any* time frames. And in the theory about the transitivity of distance relations (in Fig. 5), recursion models inferences about the spatial relations between *any* objects in a set, from representations about which two objects are adjacent. Essentially, recursion is ideal for

[13] Note that there is nothing in LP (or in Theory Toolbox) that says that events have to have these 5 arguments; so for example it would have been entirely possible to add a place argument to the event predicate if relevant.

[14] Note though that nesting events does not make sense in all situations, because the verb in the outer event imposes some restrictions on its object. The object of verbs such as "represent", "attend", "perceive", and "like" can be other events. But the object of a verb such as "perform" is usually a single behavior (and not another event).

Table 1
Construct names versus event predicates.

Construct name	Event predicate
Self esteem	`event(H, like, H, 1)`
In-group favoritism	`event(H, like, G, 1)` ⇐ `belong(H, G)`
Outgroup derogation	`event(H, like, G, 0)` ⇐ ¬`belong(H, G)`
Prejudice	`event(H1, like, H2, 0)` ⇐ `belong(H1, G1)` ∧ `belong(H2, G2)` ∧ ¬(G1 = G2)
Group acceptance	`event(G, like, H, 1)`
Self-derogation	`event(H, like, H, 0)`
Interpersonal attraction	`event(H1, like, H2, 1)` ⇐ ¬(H1 = H2)

Note. The time argument in event has been omitted for the sake of simplicity.

describing how people think about any transitive relation, such as location, causality, identity, parthood, logical implication, and so on.

2.3.4. *Universal statements about well-defined domains*

Theories consist of a set of universal statements about relations between events (e.g. Popper, 1972), i.e. they are nomothetic (law-like) descriptions. A universal statement is a statement that is true for all events of a certain kind. A singular statement, in contrast, is a statement that is true for one specific event. Singular statements are not the things that make up theories; instead, they are the things that are entailed by theories. Consider this quote from Ajzen (1991), which illustrates the distinction between universal and singular statements:

> As in the original theory of reasoned action, a central factor in the theory of planned behavior is the individual's intention to perform a given behavior. Intentions are assumed to capture the motivational factors that influence a behavior; they are indications of how hard people are willing to try, of how much of an effort they are planning to exert, in order to perform the behavior. As a general rule, the stronger the intention to engage in a behavior, the more likely should be its performance. (p. 181)

Here the aim is to communicate a universal statement. Clearly, "the individual" does not refer to John, Mary or any other specific person; instead it refers to all past, present and future individuals of a certain kind. Analogously, "the behavior" does not refer to somebody's exercise at the gym yesterday, somebody's blood donation the 20th of June 2018, or any other specific behavior; instead it refers to all past, present and future behaviors of a certain kind.

A powerful feature of LP is that it is possible to describe properties and relations that hold for all objects in one or more domains, while at the same time constructing detailed definitions of these domains. In line with our design principles this is realized by placing any number of constraints in the antecedent of a clause and by using variables. As an example, consider the clause on happiness in the emotion theory (Fig. 3). Here it is possible to explicitly say that the effect of appraising a goal congruent event on happiness holds for all events, all goals and all humans; the implication succeeds whenever the conditions in the antecedent hold. And in the theory about substance misuse (Fig. 4A and Fig. 4B), for example, it is possible to say that observational learning holds for any human, any misuse behavior, any referent, any positive outcome and any time frames (main clause 7). Note that both of these examples have rather liberal domain restrictions (e.g. they hold for any human); but we could easily have added any number of restrictions in the antecedent of each clause, such as `adult(H)`, `belongs(H, somePopulation)`, `voluntary(M)`, and so on, even domain restrictions that are probabilistic, such as `belongs(H, somePopulation, X)`, where `X` is a probability.

The point is that LP provides a mechanism for doing two things in conjunction: To say that a relation holds for *all* objects in one or more domains *and* to build detailed definitions of these domains. Given that psychological theories usually involve nomothetic statements, variables and domain specifications play an important role in constructing such representations.

2.3.5. *Modularity*

Logic programming also has important advantages in terms of modularity: It is possible to add new components to a theory without modifying the components that already exist in the theory, and the amount of information that is gained by adding new components goes beyond the amount of information just contained in the added components. Consider a simple example that illustrates the last property. Suppose that a program contains `mortal(X)` ⇐ `human(X)` and `human(socrates)`. Adding `human(plato)`. to this program does not just increment our knowledge with the fact that Plato was a human being; we also come to know that Plato was mortal.

In a logic program it is possible to add an arbitrary number of background clauses and main clauses to the ones that already exist in a theory. In the substance misuse theory (Fig. 4A and Fig. 4B), for example, it would be easy to add the background assumption that eating unhealthy is a misuse behavior and a clause about the effects of information campaigns on peoples' cognitions about the outcomes of substance misuse. Doing so simply involves adding two clauses to the program and does not change the information that already exists in the program. Also, the amount of information that is gained is higher than the amount of information added. Suppose that we add the clause `misuse(eatUnhealthy)`. Beyond incrementing our knowledge with just this information, we also come to know all distant and proximal antecedents to eating unhealthy. All main clauses in the theory have `misuse(M)` in their antecedent and this unifies with `misuse(eatUnhealthy)`. Among other things we would learn that an explanation of eating unhealthy is that somebody expects that eating unhealthy causes pleasure.

2.4. *Summary*

In this section we have discussed some positive features of LP with regard to theory representation. We argued that logic programs can encode relevant background assumptions, make exact statements about the meaning of events (constructs), make exact statements about relations between events, encode universal statements about well-defined domains, and that they are modular representations. In the next section we will present a set of tools for making scientific inferences from theory programs.

3. **Inference**

Human scientific reasoning, like any other kind of human reasoning, is imperfect in at least two ways. First, it heavily relies on working memory, which has a limited amount of capacity (Baddeley, 2012). This is problematic, because unlike everyday reasoning, which often succeeds using shortcuts, scientific reasoning should be an exhaustive search for a valid conclusion that considers everything that is relevant to a problem, no matter how complex it is. Consider the substance misuse example. This theory is actually not that large and it says some relevant things about drug abuse. Still, the number of components in the theory and the number of relations that might or might not exist between these components is much larger than what humans typically manage (in general, the number of relations that can exist between any set of components grows exponentially with the number of components). Secondly, human scientific reasoning is sometimes distorted by different biases, like belief biases and confirmation biases (e.g. Holyoak & Morrison, 2005). These can lead up to conclusions that deviate from normative models, such as logic or mathematics. And when a problem is large, biases may cause people to selectively attend to some information while ignoring other (potentially relevant) information (e.g. Koslowski, 2013).

When a theory is encoded in a logic program, it can be handed to an

```
Query
q1 ⇐    GOAL = event(_, _, _, _, _)
    ∧ INPUT = [
        source(_),
        human(somebody),
        referent(friend, somebody),
        exogenousEvent(_, _, _, _, _)
    ]
    ∧ provable(GOAL, INPUT, RESULT)
    ∧ showProvable(RESULT) ∧ fail.
```

```
Output
…
event(somebody,represent,event(somebody,like,useAmphetamine,2,1),2,A)
event(somebody,represent,event(somebody,like,useAmphetamine,3,1),3,A)
…
event(somebody,represent,event(somebody,control,useHeroin,2,1),2,A)
event(somebody,represent,event(somebody,control,useHeroin,3,1),3,A)
…
event(somebody,represent,event(somebody,should,useCocaine,2,1),2,A)
event(somebody,represent,event(somebody,should,useCocaine,3,1),3,A)
…
event(somebody,represent,event(somebody,intend,useCannabis,3,1),3,A)
event(somebody,represent,event(somebody,intend,useCannabis,4,1),4,A)
…
event(somebody,perform,useBenzodiazepines,5,A)
event(somebody,perform,useBenzodiazepines,6,A)
…
event(somebody,experience,physicalHarm,5,A)
event(somebody,experience,physicalHarm,6,A)
```

Fig. 7. Query with provable and output from substance misuse example.

algorithm that generates valid conclusions. In this section we describe the software package Theory Toolbox (Rohner, 2020). Theory Toolbox is a logic program that reasons about theory programs (as they were described in section 2. Representation). It was inspired by, and extends, some meta-interpreters for Prolog (e.g. Sterling & Shapiro, 1994; Triska, 2020). Theory Toolbox contains 6 predicates: Provable, Prove, Max Value, Min Value, Incoherence and Falsifiability. Broadly speaking, these can compute any predictions, explanations and numerical solutions that are entailed by a theory, as well as information about the quality of a theory in terms of its internal coherence and falsifiability. Prediction, explanation and theory evaluation are fundamental components of science. Theory Toolbox is made available in a theory program by writing `:-include('theoryToolbox.pl').` in the beginning of the file.

3.1. Provable

`provable(GOAL, INPUT, RESULT)` is used to determine if `GOAL` is entailed by a theory and, in case `GOAL` contains any variables, to find the variable instantiations that are entailed by the theory. Results are printed with `showProvable(RESULT)`. `GOAL` should be an atomic formula in which any argument is a constant, a variable, a function or an anonymous variable. An anonymous variable is written as a single underscore (_) and it unifies with anything. Using an anonymous variable for an argument in a `GOAL` formula is equivalent to the instruction "Return anything that might unify with this formula and argument according to the theory program.". Suppose that Provable is used on the toy example in Fig. 1. `GOAL` could contain any consequent that appears in this theory, for example `GOAL = event(h1, uses, _, _)`, which,

```
Query
q2 ⇐    GOAL = event(somebody, experience, physicalHarm, 6, _)
    ∧ misuse(MISUSE)
    ∧ INPUT = [
        source(_),
        human(somebody),
        referent(friend, somebody),
        event(somebody, perform, MISUSE, 1, 1)
    ]
    ∧ provable(GOAL, INPUT, RESULT)
    ∧ write(MISUSE) ∧ write(': ') ∧ showProvable(RESULT) ∧ fail.
```

```
Output
useHeroin: event(somebody,experience,physicalHarm,6,0.93)
useCocaine: event(somebody,experience,physicalHarm,6,0.78)
useAlchohol: event(somebody,experience,physicalHarm,6,0.1652192927)
useBenzodiazepines: event(somebody,experience,physicalHarm,6,0.057002405399999984)
useAmphetamine: event(somebody,experience,physicalHarm,6,0.12090672600000002)
useTobacco: event(somebody,experience,physicalHarm,6,0.1441274681)
useCannabis: event(somebody,experience,physicalHarm,6,0.05198477130000001)
useLSD: event(somebody,experience,physicalHarm,6,0.10791331579999999)
useEcstasy: event(somebody,experience,physicalHarm,6,0.021875)
useStreetMethadone: event(somebody,experience,physicalHarm,6,0.08035199999999999)
```

Fig. 8. Query with provable and output from substance misuse example.

```
Query
q1 ⇐    GOAL = event(somebody, deduce, plan(start, _, satisfied), time, _)
    ∧ INPUT = [
        human(somebody),
        event(somebody, represent, transition(start, openFridge, fridgeOpen), time, 1),
        event(somebody, represent, transition(start, openWaterFaucet, haveBeverage), time, 1),
        event(somebody, represent, transition(fridgeOpen, getMilk, haveBeverage), time, 1),
        event(somebody, represent, transition(fridgeOpen, getJuice, haveBeverage), time, 1),
        event(somebody, represent, transition(haveBeverage, getGlass, haveGlass), time, 1),
        event(somebody, represent, transition(haveBeverage, drinkBeverage, satisfied), time, 1),
        event(somebody, represent, transition(haveGlass, pourBeverageInGlass, beverageInGlass), time, 1),
        event(somebody, represent, transition(beverageInGlass, drinkBeverage, satisfied), time, 1)
    ]
    ∧ provable(GOAL, INPUT, RESULT)
    ∧ showProvable(RESULT) ∧ fail.
```

```
Output
event(somebody,deduce,plan(start,(openFridge,getMilk,drinkBeverage),satisfied),time,0.6400000000000001)
event(somebody,deduce,plan(start,(openFridge,getMilk,getGlass,pourBeverageInGlass,drinkBeverage),satisfied),time,0
.40960000000000013)
event(somebody,deduce,plan(start,(openFridge,getJuice,drinkBeverage),satisfied),time,0.6400000000000001)
event(somebody,deduce,plan(start,(openFridge,getJuice,getGlass,pourBeverageInGlass,drinkBeverage),satisfied),time,
0.40960000000000013)
event(somebody,deduce,plan(start,(openWaterFaucet,drinkBeverage),satisfied),time,0.8)
event(somebody,deduce,plan(start,(openWaterFaucet,getGlass,pourBeverageInGlass,drinkBeverage),satisfied),time,0.51
20000000000001)
```

Fig. 9. Query with provable and output from planning example.

given how the theory was defined, means "What substance does h1 use with what probability?" (substance was the third argument and probability was the fourth argument). One of the results would then be `event(h1, uses, heroin, 0.81)`. With `GOAL = event(h1, uses, heroin, 0.81)`, instead, the result would be the word `true`, meaning that it is true that this goal is provable from the theory program (a goal that is not provable generates `false`).

The second argument that should be defined is `INPUT`. `INPUT` is used to inform Provable about anything that is considered to be provable, *in addition* to the information in a theory program. In the section on proof search we said that a goal is provable in each one of these cases: (1) the goal is true; (2) the goal unifies with a consequent that has an empty antecedent; and (3) the goal unifies with a consequent in which the antecedent is provable. With `provable(GOAL, INPUT, RESULT)` a goal is provable in an additional case: (4) The goal unifies with an atom that appears in `INPUT`. The `INPUT` argument is a mechanism for temporarily asserting information about any specific instance to which a theory is applied. This information does not belong *in* the theory; instead, it is *handed to* the theory in order to derive conclusions from it. Theories, per se, should only make general statements, and should not be filled with all the particular instances that are within their scope. Consider the toy example in Fig. 1. As it stands, this is not a good theory representation: The clauses on lines 3, 4, 11 and 12 should be removed, because these just represent particular cases to which the theory can be applied. Instead, this information belongs in `INPUT`. Syntactically, `INPUT` is a Prolog list, which is a comma delimited enumeration of any number of atoms in square braces. After amending the toy example, Provable could thus be handed `INPUT = [human(h1), event(h1, used, heroin, 0.90)]`. The statement means that `human(h1)` and `event(h1, used, heroin, 0.90)` are provable (in addition to the information in the theory program).

Note that the examples in Figs. 2, 3 and 4A, 4B, 5 and 6 contain a commented section named "INPUT". This section lists all the atoms that are required in `INPUT` in order for all the consequents of the respective theory to be provable. When designing a theory, it is good practice to inform third parties about any assumptions in a separate section in the beginning of the file. In this way it is possible to quickly get a picture of what has to be defined in `INPUT` to get the intended output, without having to look through all the clauses of the theory. More examples of how `INPUT` is used are shown below.

Consider the query and output in Fig. 7 which is based on the substance misuse example (Fig. 4A and Fig. 4B). It represents a very broad search for any conclusions that are entailed by the theory, a good starting point (nothing that the theory entails is overlooked). The first line in the query states that the goal is `event(_, _, _, _, _)`. This is the most general goal with respect to the main clauses in the theory, because all of them have the five argument event predicate as a consequent, and because all arguments in the goal are anonymous variables (no argument in the goal will fail to unify with the corresponding argument in the consequent of the theory). `INPUT` then defines the atoms that are specified in the INPUT section of the theory. Note that there is an atom denoted `exogenousEvent` in which all arguments are anonymous variables. This renders all 6 `exogenousEvent` atoms in main clauses 1, 2 and 3 provable, and, with the other information in INPUT, the consequents of main clauses 1, 2, 3, 4, 5, 6, 7 and 8 (technically speaking, the meta interpreter in Provable has `true` as a base case, so this is necessary). Using anonymous variables in this way yields a result in which variables that are not bound to constants in the theory will remain variables, and in which variables that are bound to constants in the theory will be constants (see this footnote[15] for a detailed explanation and example). Finally, `showProvable(RESULT) ∧ fail` is used to print to the console and to find all solutions to the goal,

[15] By using anonymous variables in `INPUT` it is possible to obtain conclusions that involve unbound variables, as well as conclusions that involve constants that are entailed by a theory program, per se. Consider for example what happens with the last (value) argument in `exogenousEvent`, which is an anonymous variable in `INPUT`. Among others, it will unify with the `X2` variable in the first main clause of the theory in Fig. 4B. But because the values of `X2` and `X3` are unknown, the value of `X1` will be unknown as well (as will all other values that depend on `X1`). So in this case, the output will just be a variable (i.e. that the probability of an event is unknown). Compare this to what happens when an anonymous variable is unified with, say, the `T2` variable in the first main clause. In this case, `T2` has at least one grounding in the theory, because `precedes(T2, T1)` is provable with `T2 = 1` and so on. So this argument will be a constant in any conclusions that are derived from this `exogenousEvent`. The same goes for, say, the `M` variable in this clause, which will be bound to `useHeroin`, and so on. By using anonymous variables in this way, it is possible to find out if a theory entails a certain event, independently of whether some of its arguments are unknown (i.e. variables), or known (i.e. constants).

respectively (`fail` is a built-in Prolog predicate).

The output from this query generates several results; only some of these are shown in Fig. 7. The last result, for example, means that it is provable that somebody experiences physical harm with probability *A* in time frame 6. The value of *A* is unknown because no probability values were handed to Provable, so a unique solution to the system of equations in the theory does not exist. Note, though, that knowing that harm occurs with an unknown probability is distinct from knowing nothing at all about harm (in the first case, the theory contains an equation for estimating the probability of harm, in the second case it doesn't).

Fig. 8 shows a more specific query on the substance misuse example. The goal `event(somebody, experience, physicalHarm, 6, _)` means "What is the probability that somebody experiences physical harm in time frame 6?". The second line in this query unifies the variable `MISUSE` with any misuse behavior (e.g. `useHeroin`); `MISUSE` is then included in `INPUT` to assume that somebody performs this behavior in time frame 1 with a probability of 1. Finally, `write(MISUSE)` and `showProvable(RESULT)` are used to print to the console (`write` is a built-in Prolog predicate). Note, for example, that the probability of harm is higher for alcohol use than for benzodiazepine use even though the background assumptions in the theory stated that alcohol is less harmful (in Fig. 4A). This is because alcohol is rated to produce more pleasure which, by the clause about positive reinforcement, increases the probability of misuse across time frames, and because the probability of experiencing harm is weighted with the probability of misuse.

Fig. 9 illustrates how Provable and Show Provable can be applied to the planning example (in Fig. 6). Note that plans vary in length and that the probability of deducing a longer plan is lower than the probability of deducing a shorter plan. This is because the probability of deducing a plan is weighted by 0.8 in each recursion in the second clause of the theory.

3.2. Prove

`prove(GOAL, INPUT, PROOF)` generates a `PROOF` for `GOAL` given `INPUT` (or `false` if `GOAL` is not provable) .[16] The predicate is used in conjunction with `showProof(PROOF)` to display a proof in readable form. Essentially, these predicates return explanations for *why* a certain conclusion is entailed by a theory. Show Proof displays the sub-proof for a goal below the goal, indented to the right, where sub-goals with the same level of indentation come from the same clause. An important property of Prove is that it can generate a numerical solution for a goal that involves a mathematical expression, if a unique solution exists, but also generate a symbolic solution, when a unique solution does not exist. This is useful when the probabilities of one or more antecedent events are unknown but a user still wants to get an explanation from a theory.

Fig. 10 contains an example of using Prove on the substance misuse example (in Fig. 4A and Fig. 4B). The goal `event(somebody, perform, useTobacco, 6, _)` means "The unknown probability of using tobacco in time frame 6.". Note that `INPUT` defines the atoms listed in the input section, including an atom `exogenousEvent` with all arguments set to anonymous variables. This is necessary for the consequents of main clauses 1, 2 and 3 to be provable, and in turn the consequents of main clauses 4, 5, 6, 7, and 8. Prove will only generate proofs that involve provable antecedents, so to obtain a complete explanation from a theory, all of its antecedents have to hold (the meta interpreter in Prove has `true` as a base case, like Provable). Using an input with anonymous variables will also generate proofs in which variables that are not bound to constants in the theory will remain variables, and in which variables that can be bound to constants in the theory will be constants (again, see footnote 15 for an explanation and example).

The lower part of the figure shows one of the longer proofs for the goal. Prove returns a symbolic solution because no numbers were handed to it in `INPUT`; variables with unknown values are labelled alphabetically, starting with `A`. Note that the antecedent `true` either occurs when there is a clause with an empty antecedent in the theory program or when an atom unifies with a term in `INPUT` (which is passed to Prove). In Fig. 10 we have omitted the proofs for `precedes` for space reasons.

Fig. 11 shows an example in which Prove and Show Proof are applied to the emotion theory (in Fig. 3). The goal `event(h1, experience, _, present, _)` means "What does h1 experience in the present with what probability?". On the second line in the query the variable `T` is unified with `past` or `future` to determine the effects of having emotion eliciting events in the past or future (note where `T` occurs in `INPUT`). Fig. 11 only shows two proofs for when `T = past` because of space limitations. When `T = future`, the fear solution is the only one that exists (other emotions were defined to occur in response to events in the past in the theory).

3.3. Max Value

`maxValue(X, GOAL, INPUT)` and `showMaxValue(GOAL, INPUT)` find and show a proof for `GOAL` given `INPUT`, such that the numerical argument `X` in `GOAL` is as high as possible (internally Show Max Value calls Prove, which was explained in the previous section). If there is more than one proof for attaining the maximum value, all proofs are returned. If `GOAL` is not provable the output is `false`. These predicates thus play an important role for finding a solution to a system of equations with several unknowns. `INPUT` should define a couple of alternative values for numerical variables that are exogenous; i.e. variables that do not appear as dependents in any theory equation. In concrete terms, suppose that some parts of a theory are `{X1 = X2 - 0.5 * X3}` and `{X3 = X4 + X5}`. Here, the values of `X2`, `X4` and `X5` should be defined in `INPUT`, because they are exogenous. If the assumption is that `X2`, `X4` and `X5` each can take on the values `0.1` or `0.9`, the maximum of `X1` is `0.8` with `X2 = 0.9`, `X4 = 0.1` and `X5 = 0.1`. How many values and which values are chosen for exogenous variables is up to the user. Choosing fewer values speeds up the search but increases the risk of missing a global maximum; choosing more values slows the search but reduces the risk of missing a global maximum. The algorithm in Max Value simply iterates through all solutions and picks the one with the highest value (in the future we will probably implement a better optimization algorithm).

Broadly speaking, Max Value and Show Max Value can be used to find the antecedent conditions of a problem, with a goal such as `event(somebody, perform, useHeroin, 6, X)`, or how a positive outcome can be achieved, with a goal such as `event(somebody, perform, healthScreenings, 6, X)`.

Fig. 12 shows an example in which Max Value and Show Max Value are applied to the substance misuse theory (in Fig. 4A and Fig. 4B). The query finds a proof for `event(somebody, perform, eatUnhealthy, 6, X)` such that the value of `X` is as high as possible. In `INPUT` three alternative probabilities are given to the exogenous events in the theory. The other arguments are set to anonymous variables to ensure that all direct and indirect antecedents of the goal are provable and therefore returned in the proof, independently of whether they involve unbound variables or constants entailed by the theory program (see footnote 15 for an explanation). Note that an atom about eating unhealthy has been added in `INPUT`. By doing so the theory generalizes to eating unhealthy even though information about this behavior is missing from its background assumptions (see Fig. 4A). This highlights how easy it is to extend a theory so that it generalizes to additional instances. The lower part of Fig. 12 shows one of the proofs. In the output, note how the probabilities of exogenous events maximize the probability of the goal event (eating unhealthy), and that the probabilities of exogenous events propagate in the system of equations that lead up to the probability of the goal.

[16] A special thanks goes to Daniel Lyons on Stack Overflow for pointing out the right direction when approaching this problem.

```
Query
q3 ⇐    GOAL = event(somebody, perform, useTobacco, 6, _)
    ∧ INPUT = [
        source(_),
        human(somebody), human(friend),
        referent(friend, somebody), referent(somebody, friend),
        exogenousEvent(_, _, _, _, _)
    ]
    ∧ prove(GOAL, INPUT, PROOF)
    ∧ showProof(PROOF).
```

```
Output
PROOF
event(somebody,perform,useTobacco,6,A) ⇐
     source(operantLearning) ⇐ true
     human(somebody) ⇐ true
     misuse(useTobacco) ⇐ true
     outcome(pleasure) ⇐ true
     positive(pleasure) ⇐ true
     reinforcer(pleasure) ⇐ true
     precedes(5,6) ⇐true
     event(useTobacco,cause,pleasure,5,0.77) ⇐ true
     event(somebody,perform,useTobacco,5,B) ⇐
          source(vicariousLearning) ⇐ true
          human(somebody) ⇐ true
          misuse(useTobacco) ⇐ true
          referent(friend,somebody) ⇐ true
          outcome(pleasure) ⇐ true
          positive(pleasure) ⇐ true
          reinforcer(pleasure) ⇐ true
          precedes(4,5) ⇐ true
          event(useTobacco,cause,pleasure,4,0.77) ⇐true
          event(friend,perform,useTobacco,4,C) ⇐
               source(theoryOfPlannedBehavior) ⇐ true
               human(friend) ⇐ true
               misuse(useTobacco) ⇐ true
               precedes(3,4) ⇐true
               event(friend,represent,event(friend,intend,useTobacco,3,1),3,D) ⇐
                    source(theoryOfPlannedBehavior) ⇐ true
                    human(friend) ⇐ true
                    misuse(useTobacco) ⇐ true
                    precedes(2,3) ⇐true
                    event(friend,represent,event(friend,like,useTobacco,2,1),2,E) ⇐
                         source(theoryOfPlannedBehavior) ⇐ true
                         human(friend) ⇐ true
                         misuse(useTobacco) ⇐ true
                         outcome(pleasure) ⇐ true
                         outcome(physicalHarm) ⇐ true
                         positive(pleasure) ⇐ true
                         negative(physicalHarm) ⇐ true
                         precedes(1,2) ⇐true
                         exogenousEvent(friend,represent,event(useTobacco,cause,pleasure,1,1),1,F) ⇐ true
                         exogenousEvent(friend,represent,event(useTobacco,cause,physicalHarm,1,1),1,G) ⇐ true
                         {E=F*(1-G)} ⇐ true
                    event(friend,represent,event(friend,control,useTobacco,2,1),2,H) ⇐
                         source(theoryOfPlannedBehavior) ⇐ true
                         human(friend) ⇐ true
                         misuse(useTobacco) ⇐ true
                         precedes(1,2) ⇐true
                         exogenousEvent(friend,represent,event(environment,afford,useTobacco,1,1),1,I) ⇐ true
                         exogenousEvent(friend,represent,event(friend,affect,environment,1,1),1,J) ⇐ true
                         {H=I*J} ⇐ true
                    event(friend,represent,event(friend,should,useTobacco,2,1),2,K) ⇐
                         source(theoryOfPlannedBehavior) ⇐ true
                         human(friend) ⇐ true
                         referent(somebody,friend) ⇐ true
                         misuse(useTobacco) ⇐ true
                         precedes(1,2) ⇐true
                         exogenousEvent(somebody,represent,event(friend,should,useTobacco,1,1),1,L) ⇐ true
                         exogenousEvent(friend,represent,event(friend,comply,somebody,1,1),1,M) ⇐ true
                         {K=L*M} ⇐ true
                    {D=E*H*K} ⇐ true
               {C=D} ⇐ true
          exogenousEvent(somebody,attend,friend,4,N) ⇐ true
          exogenousEvent(somebody,capable,useTobacco,4,O) ⇐ true
          {B=0.77*C*N*O} ⇐ true
     {A=0.77*B} ⇐ true
```

Fig. 10. Query with prove and output from substance misuse example.

3.4. Min Value

`minValue(X,GOAL,INPUT)` and `showMinValue(GOAL,INPUT)` work in the same way as the previous predicates, except that they find and show a proof for `GOAL` given `INPUT`, such that the numerical argument `X` in `GOAL` is as low as possible. This can come in handy when answering questions such as how a problem can be minimized, with a goal like `event(somebody, perform, eatUnhealthy, 6, X)`, or why the current state of affairs differs from a desired state, with a goal like `event(somebody, perform, eatHealthy, 6, X)`.

```
Query
q2 ⇐    GOAL = event(h1, experience, _, present, _)
    ∧ (T = past ∨ T = future)
    ∧ INPUT = [
        human(h1), human(h2),
        event(jobLoss),
        event(h1, appraise, event(h1, experience, jobLoss, T, 1), present, 0.9),
        event(h1, appraise, event(jobLoss, congruent, achievement, T, 1), present, 0.1),
        event(h1, appraise, event(h1, cause, jobLoss, T, 1), present, 0.9),
        event(h1, appraise, event(h2, cause, jobLoss, T, 1), present, 0.1),
        event(h1, appraise, event(world, cause, jobLoss, T, 1), present, 0.1)
    ]
    ∧ prove(GOAL, INPUT, PROOF) ∧ showProof(PROOF) ∧ fail.
```

```
Output
PROOF
event(h1,experience,anger,present,0.004374) ⇐
     human(h1) ⇐ true
     human(h2) ⇐ true
     ¬h1=h2 ⇐ true
     event(jobLoss) ⇐ true
     goal(achievement) ⇐ true
     precedes(past,present) ⇐ true
     event(h1,value,achievement,present,0.6) ⇐
          human(h1) ⇐ true
          time(present) ⇐ true
     event(h1,appraise,event(h1,experience,jobLoss,past,1),present,0.9) ⇐ true
     event(h1,appraise,event(jobLoss,congruent,achievement,past,1),present,0.1) ⇐ true
     event(h1,appraise,event(h1,cause,jobLoss,past,1),present,0.9) ⇐ true
     event(h1,appraise,event(h2,cause,jobLoss,past,1),present,0.1) ⇐ true
     event(h1,appraise,event(world,cause,jobLoss,past,1),present,0.1) ⇐ true
     {0.004374=0.6*0.9*(1-0.1)*(1-0.9)*0.1*(1-0.1)} ⇐ true

PROOF
event(h1,experience,shame,present,0.35429400000000005) ⇐
     human(h1) ⇐ true
     human(h2) ⇐ true
     ¬h1=h2 ⇐ true
     event(jobLoss) ⇐ true
     goal(achievement) ⇐ true
     precedes(past,present) ⇐ true
     event(h1,value,achievement,present,0.6) ⇐
          human(h1) ⇐ true
          time(present) ⇐ true
     event(h1,appraise,event(h1,experience,jobLoss,past,1),present,0.9) ⇐ true
     event(h1,appraise,event(jobLoss,congruent,achievement,past,1),present,0.1) ⇐ true
     event(h1,appraise,event(h1,cause,jobLoss,past,1),present,0.9) ⇐ true
     event(h1,appraise,event(h2,cause,jobLoss,past,1),present,0.1) ⇐ true
     event(h1,appraise,event(world,cause,jobLoss,past,1),present,0.1) ⇐ true
     {0.35429400000000005=0.6*0.9*(1-0.1)*0.9*(1-0.1)*(1-0.1)} ⇐ true
…
PROOF
event(h1,experience,fear,present,0.48600000000000004) ⇐
     human(h1) ⇐ true
     event(jobLoss) ⇐ true
     goal(achievement) ⇐ true
     precedes(present,future) ⇐ true
     event(h1,value,achievement,present,0.6) ⇐
          human(h1) ⇐ true
          time(present) ⇐ true
     event(h1,appraise,event(h1,experience,jobLoss,future,1),present,0.9) ⇐ true
     event(h1,appraise,event(jobLoss,congruent,achievement,future,1),present,0.1) ⇐ true
     {0.48600000000000004=0.6*0.9*(1-0.1)} ⇐ true
```

Fig. 11. Query with prove and output from emotion example.

3.5. *Incoherence*

`incoherence(INPUT, GOAL1, GOAL2, THRESHOLD, X1, X2)` determines if a theory entails that `GOAL1` and `GOAL2` differ in their numerical variables `X1` and `X2` (respectively) more than the number[17] in `THRESHOLD;` specifically if `abs(X1 - X2)` > `THRESHOLD` (absence of incoherence results in the output `false`). Suppose that there are two goals that have the same variables as arguments except for their probabilities; e.g. `GOAL1` = `event(S, V, O, T, X1)` and `GOAL2` = `event(S, V, O, T, X2)`. When these goals appear in the same clause, each variable `S`, `V`, `O`, `T` has to hold the same constant because the scope of a variable is the clause. If a theory entails that `X1` and `X2` are different, it means that the theory makes incoherent predictions. That is, even when both events have the same semantic content (the same variables in the subject, verb, object and time arguments) they have different probabilities. This could mean that there is a problem in the theory.

Consider Fig. 13 where Incoherence and the corresponding output predicate Show Incoherence are applied to the theory of simple phobia (in Fig. 2). Is this theory coherent? It seems not, because it predicts different probabilities of experiencing fear even if the input is the same (why the incoherence occurs is shown in the output). The theory should be amended somehow, e.g. by making encounters of a phobic object and avoidance of the object asynchronous, or by removing the first main clause in Fig. 2). Note that incoherence only arises when `source(plausibleAssumption)` is assumed to hold; removing it from `INPUT` does not result in an incoherent theory.

3.6. *Falsifiability*

`falsifiability(GOAL, INPUT, N)` counts the number of unique consequents `N` (i.e. predictions) with respect to `GOAL` given `INPUT`. Results are printed to the console by using

[17] It is up to the user to choose an appropriate value for `THRESHOLD`.

Query

```
q5 ⇐    GOAL = event(somebody, perform, eatUnhealthy, 6, X)
    ∧ INPUT = [
        source(_),
        misuse(eatUnhealthy),
        human(somebody), human(friend),
        referent(friend, somebody), referent(somebody, friend),
        exogenousEvent(_, _, _, _, 0.1),
        exogenousEvent(_, _, _, _, 0.5),
        exogenousEvent(_, _, _, _, 0.9)
    ]
    ∧ maxValue(X, GOAL, INPUT)
    ∧ showMaxValue(GOAL, INPUT).
```

Output

```
MAX VALUE (PROOF)
event(somebody,perform,eatUnhealthy,6,0.5314410000000002) ⇐
     source(theoryOfPlannedBehavior) ⇐ true
     human(somebody) ⇐ true
     misuse(eatUnhealthy) ⇐ true
     precedes(5,6) ⇐
          time(5) ⇐ true
          time(6) ⇐ true
          {6=5+1} ⇐ true
     event(somebody,represent,event(somebody,intend,eatUnhealthy,5,1),5,0.5314410000000002) ⇐
          source(theoryOfPlannedBehavior) ⇐ true
          human(somebody) ⇐ true
          misuse(eatUnhealthy) ⇐ true
          precedes(4,5) ⇐
               time(4) ⇐ true
               time(5) ⇐ true
               {5=4+1} ⇐ true
          event(somebody,represent,event(somebody,like,eatUnhealthy,4,1),4,0.81) ⇐
               source(theoryOfPlannedBehavior) ⇐ true
               human(somebody) ⇐ true
               misuse(eatUnhealthy) ⇐ true
               outcome(pleasure) ⇐ true
               outcome(physicalHarm) ⇐ true
               positive(pleasure) ⇐ true
               negative(physicalHarm) ⇐ true
               precedes(3,4) ⇐
                    time(3) ⇐ true
                    time(4) ⇐ true
                    {4=3+1} ⇐ true
               exogenousEvent(somebody,represent,event(eatUnhealthy,cause,pleasure,3,1),3,0.9) ⇐ true
               exogenousEvent(somebody,represent,event(eatUnhealthy,cause,physicalHarm,3,1),3,0.1) ⇐ true
               {0.81=0.9*(1-0.1)} ⇐ true
          event(somebody,represent,event(somebody,control,eatUnhealthy,4,1),4,0.81) ⇐
               source(theoryOfPlannedBehavior) ⇐ true
               human(somebody) ⇐ true
               misuse(eatUnhealthy) ⇐ true
               precedes(3,4) ⇐
                    time(3) ⇐ true
                    time(4) ⇐ true
                    {4=3+1} ⇐ true
               exogenousEvent(somebody,represent,event(environment,afford,eatUnhealthy,3,1),3,0.9) ⇐ true
               exogenousEvent(somebody,represent,event(somebody,affect,environment,3,1),3,0.9) ⇐ true
               {0.81=0.9*0.9} ⇐ true
          event(somebody,represent,event(somebody,should,eatUnhealthy,4,1),4,0.81) ⇐
               source(theoryOfPlannedBehavior) ⇐ true
               human(somebody) ⇐ true
               referent(friend,somebody) ⇐ true
               misuse(eatUnhealthy) ⇐ true
               precedes(3,4) ⇐
                    time(3) ⇐ true
                    time(4) ⇐ true
                    {4=3+1} ⇐ true
               exogenousEvent(friend,represent,event(somebody,should,eatUnhealthy,3,1),3,0.9) ⇐ true
               exogenousEvent(somebody,represent,event(somebody,comply,friend,3,1),3,0.9) ⇐ true
               {0.81=0.9*0.9} ⇐ true
          {0.5314410000000002=0.81*0.81*0.81} ⇐ true
     {0.5314410000000002=0.5314410000000002} ⇐ true
```

Fig. 12. Query with max value and output from substance misuse example.

`showFalsifiability(GOAL, INPUT, N)`. Consider the theory about distance relations in Fig. 5. It contains two sub-theories. Main clause 2 is recursive: It means that somebody can deduce that *A* is beyond *C*, if that person represents that *A* is beyond *B* and can deduce that *B* is beyond *C*. Main clause 3 is not recursive: It means that somebody can deduce that *A* is beyond *C*, if that person represents that *A* is beyond *B* and represents that *B* is beyond *C*. Fig. 14 shows a query with Falsifiability and the associated output predicate. Note that `INPUT` states that the recursive sub-theory holds. As shown in the figure, this theory generates 28 predictions. Replacing `source(recursive)` with `source(non-recursive)` in `INPUT` only results in 13 predictions (not shown in the figure). The recursive sub-theory is therefore a better theory because it is more falsifiable (i.e. more general), even if it contains about the same amount of code as the non-recursive one.

3.7. Summary

In this section we have shown how the predicates in Theory Toolbox can be used to find the logical consequences of a theory (Provable), explain these consequences in terms of their antecedents (Prove), find the antecedents that maximize or minimize the probability of a consequent (Max Value and Min Value) and to evaluate a theory in terms of its internal coherence and falsifiability (Incoherence and Falsifiability). The GitHub repository for Theory Toolbox contains additional examples

```
Query
q3 ⇐    GOAL1 = event(S, V, O, T, X1)
    ∧ GOAL2 = event(S, V, O, T, X2)
    ∧ INPUT = [
        source(dsm5),
        source(plausibleAssumption),
        human(somebody),
        event(somebody, phobia, _, time, 1)
    ]
    ∧ THRESHOLD = 0.1
    ∧ incoherence(INPUT, GOAL1, GOAL2, THRESHOLD, X1, X2)
    ∧ showIncoherence(INPUT, GOAL1, GOAL2, THRESHOLD) ∧ fail.
```

```
Output
INCOHERENCE

Given inputs:
source(dsm5)
source(plausibleAssumption)
human(somebody)
event(somebody,phobia,A,time,1)

These goals are incoherent at the threshold 0.1.

PROOFS

event(somebody,fear,spiders,time,1.0) ⇐
     source(dsm5) ⇐ true
     human(somebody) ⇐ true
     object(spiders) ⇐ true
     event(somebody,phobia,spiders,time,1) ⇐ true
     {1.0=1} ⇐ true

event(somebody,fear,spiders,time,0.0) ⇐
     source(dsm5) ⇐ true
     human(somebody) ⇐ true
     object(spiders) ⇐ true
     event(somebody,phobia,spiders,time,1) ⇐ true
     event(somebody,encounter,spiders,time,0.0) ⇐
          source(plausibleAssumption) ⇐ true
          human(somebody) ⇐ true
          object(spiders) ⇐ true
          event(somebody,avoid,spiders,time,1.0) ⇐
               source(dsm5) ⇐ true
               human(somebody) ⇐ true
               object(spiders) ⇐ true
               event(somebody,phobia,spiders,time,1) ⇐ true
               {1.0=1} ⇐ true
          {0.0=1-1.0} ⇐ true
     {0.0=1*0.0} ⇐ true
```

Fig. 13. Query with incoherence and output from phobia example.

that show how the predicates in the toolbox can be used, as well as instructions for installing and running it along with SWI Prolog.

4. Discussion

In this paper, we have described how theories can be represented as logic programs and discussed the advantages in terms of accuracy and expressiveness. We also introduced a software package, Theory Toolbox, which can be used to derive predictions, derive explanations, and evaluate theories with regard to their internal coherence and falsifiability. We believe that first order logic is an important addition to the theoretical and analytical tools that are currently available in psychology, and hope that this paper will inspire researchers to construct and use theories that have this form.

There are several formal approaches to theory construction besides LP; we mentioned a few of them in the introduction. So what approach should be chosen for representing a given theory? Even if this is a complicated issue that we cannot fully resolve here, we will try to provide some tentative recommendations about when LP can be useful. These are as follows: (1) A theory is explicit, in the sense that it can easily be described in terms of words and equations; (2) A theory contains large amounts of (verbal) semantic information; (3) A theory features complex relations, that involve several parameters and perhaps nested relations (like in higher order theory of mind); (4) It is important that any conclusions that are derived from a theory can be explained to and understood by the user.

4.1. *Limitations*

We have not gone into the details of *how* theory programs are constructed from empirical data. Generally speaking, there seem to be two possible options for tackling this task. First, it is possible to work inductively, starting with data and proceeding to theory building. In this case, one or more meta-analyses seem to be the best starting point, because meta-analyses typically describe concepts and relations between the objects they reference in more or less general (abstract) terms, as first order logic does. Logic programming has a very natural way of representing relations at different levels of abstraction: More abstract relations have fewer domain restrictions in their antecedent, e.g. just `human(H)`; less abstract relations have more domain restrictions in their antecedent; e.g. `human(H)` and `adult(H)`, and so on. The second option for relating theories to data is to work deductively, by starting with theory construction and moving on to prediction and experimentation. Predicates like Provable and Prove in Theory Toolbox should play an important role in this context. In practice, of course, relating theory and data involves both induction and deduction in a continuous interplay.

We have discussed LP from a purely theoretical perspective, with little attention to its practical applications (beyond the practice of conducting science). One potential application consists of expert systems; i.e. computer programs that represent a domain of knowledge, such as the DSM5, which a person, such as a psychologist, can query in various ways. The use of Prolog for building expert systems is well known (e.g. Merritt, 1989). Our view is that LP is well suited for representing explicit and relatively narrow knowledge domains. So an

```
Query
q1 ⇐    GOAL = event(somebody, deduce, relation(_, beyond, _), time, _)
    ∧ INPUT = [
        source(recursive),
        human(somebody),
        event(somebody, represent, relation(earth, beyond, venus), time, 1),
        event(somebody, represent, relation(mars, beyond, earth), time, 1),
        event(somebody, represent, relation(jupiter, beyond, mars), time, 1),
        event(somebody, represent, relation(saturn, beyond, jupiter), time, 1),
        event(somebody, represent, relation(uranus, beyond, saturn), time, 1),
        event(somebody, represent, relation(neptune, beyond, uranus), time, 1),
        event(somebody, represent, relation(pluto, beyond, neptune), time, 1)
    ]
    ∧ falsifiability(GOAL, INPUT, N)
    ∧ showFalsifiability(GOAL, INPUT, N).
```

```
Output
FALSIFIABILITY

Given inputs:
source(recursive)
human(somebody)
event(somebody,represent,relation(earth,beyond,venus),time,1)
event(somebody,represent,relation(mars,beyond,earth),time,1)
event(somebody,represent,relation(jupiter,beyond,mars),time,1)
event(somebody,represent,relation(saturn,beyond,jupiter),time,1)
event(somebody,represent,relation(uranus,beyond,saturn),time,1)
event(somebody,represent,relation(neptune,beyond,uranus),time,1)
event(somebody,represent,relation(pluto,beyond,neptune),time,1)

There are 28 predictions for the goal event(somebody,deduce,relation(A,beyond,B),time,C).

These are:
event(somebody,deduce,relation(earth,beyond,venus),time,1.0)
event(somebody,deduce,relation(mars,beyond,earth),time,1.0)
event(somebody,deduce,relation(jupiter,beyond,mars),time,1.0)
event(somebody,deduce,relation(saturn,beyond,jupiter),time,1.0)
event(somebody,deduce,relation(uranus,beyond,saturn),time,1.0)
event(somebody,deduce,relation(neptune,beyond,uranus),time,1.0)
event(somebody,deduce,relation(pluto,beyond,neptune),time,1.0)
event(somebody,deduce,relation(mars,beyond,venus),time,0.8)
event(somebody,deduce,relation(jupiter,beyond,earth),time,0.8)
event(somebody,deduce,relation(jupiter,beyond,venus),time,0.6400000000000001)
event(somebody,deduce,relation(saturn,beyond,mars),time,0.8)
event(somebody,deduce,relation(saturn,beyond,earth),time,0.6400000000000001)
event(somebody,deduce,relation(saturn,beyond,venus),time,0.5120000000000001)
event(somebody,deduce,relation(uranus,beyond,jupiter),time,0.8)
event(somebody,deduce,relation(uranus,beyond,mars),time,0.6400000000000001)
event(somebody,deduce,relation(uranus,beyond,earth),time,0.5120000000000001)
event(somebody,deduce,relation(uranus,beyond,venus),time,0.40960000000000013)
event(somebody,deduce,relation(neptune,beyond,saturn),time,0.8)
event(somebody,deduce,relation(neptune,beyond,jupiter),time,0.6400000000000001)
event(somebody,deduce,relation(neptune,beyond,mars),time,0.5120000000000001)
event(somebody,deduce,relation(neptune,beyond,earth),time,0.40960000000000013)
event(somebody,deduce,relation(neptune,beyond,venus),time,0.32768000000000014)
event(somebody,deduce,relation(pluto,beyond,uranus),time,0.8)
event(somebody,deduce,relation(pluto,beyond,saturn),time,0.6400000000000001)
event(somebody,deduce,relation(pluto,beyond,jupiter),time,0.5120000000000001)
event(somebody,deduce,relation(pluto,beyond,mars),time,0.40960000000000013)
event(somebody,deduce,relation(pluto,beyond,earth),time,0.32768000000000014)
event(somebody,deduce,relation(pluto,beyond,venus),time,0.2621440000000001)
```

Fig. 14. Query with falsifiability and output from distance example.

expert system for the whole of psychology seems difficult to implement, but LP ontologies for domains such as the DSM5, psychotherapy recommendations, modern personality theory, or memory theory, to name a few, are completely within reach. Logic programs for such domains could then be complemented with suitable user interfaces to enable clinicians, students or the general public, to pose queries and get answers. Interestingly, SWI Prolog plays extremely well with the web, so it would not be too hard to implement both back-end and front-end for an expert system completely in SWI Prolog.

In this context, a potential limitation of LP is that it may be difficult to cover *all* the information in a knowledge domain in a theory program, especially when the domain involves large amounts of tacit knowledge. Consider a concrete example. Suppose than an emotion theory says that negative emotions arise when a person experiences a goal incongruent event. When reading this, most people can deduce that events that threaten survival will cause negative emotions, because it is common knowledge that survival is an important goal. But the emotion theory in Fig. 3 does not contain this information so it just yields the blunt conclusion "not provable". Still, there are ways to reduce the problem. One is to use different knowledge elicitation techniques such as expert interviews and "think aloud" protocols (see for example Gavrilova & Andreeva, 2012). Another is to base programs on systematic literature reviews and meta-analyses, as mentioned earlier. Of course, such techniques do not guarantee that a theory program will cover everything, but they are at least practical steps in the right direction. We also want to reiterate that the aim of this project is to devise a scheme for theory representation and inference, not to build something that replaces human experts. This means that the relevant comparisons for our LP approach are other knowledge representation methods such as natural language and mathematics (not humans). Accordingly, the problem with missing tacit knowledge is not exclusive to LP. For example, there are many texts about emotion, written in natural language, that do not explicitly say that survival is an important goal.

From a more technical perspective, there is a limitation that merits some attention. Our approach to representing and computing probability relies on CLPR constraints that are placed in the clauses of a theory, and not on a specialized probabilistic logic programming system. Using CLPR in this way is simple and straight forward, because all equations are stated explicitly *in* the theory and therefore transparent to (and in control of) the user: There are no assumptions and/or equations "behind the curtains" that affect the numerical output. Simplicity and transparency where important priorities. In addition, we found that a CLPR representation of probability (in the theory), was most practical to work with while we experimented with the meta-interpreter in Prove. The drawback of this approach is that it lacks certain advanced features that are present in systems for probabilistic logic programming, out of the box, such as CPLINT and PROBLOG (De Raedt et al., 2007; Riguzzi, 2018). CPLINT, for example, can be used to perform both exact inference and approximate inference (with Monte Carlo sampling), and to learn the parameters as well as the structure of probabilistic logic programs (Riguzzi, 2018).

Finally, we should mention two pitfalls in Prolog (a detailed discussion, however, is beyond our scope; instead we refer the interested reader to Nilsson & Maluszynski, 1995; Shapiro & Sterling, 1986). The first pitfall is that some (logically correct) recursive programs do not terminate (i.e. they "loop" forever) because of how atoms are ordered in an antecedent. But this problem is easily overcome by rearranging the order of atoms of one or more clauses (placing a call to the consequent of a recursive clause last in the antecedent); adding tabling to a recursive clause can also help.[18] The second pitfall is that negation is unsound in some cases: When a negated atom contains one or more unbound variables; the ensuing conclusions are not logically correct. Here as well, there is a solution: To make sure that all variables in a negated atom are bound to constants before the negation.

4.2. Future research

In future work, there seem to be at least three goals that are worth pursuing. First, it would be interesting to explore how Theory Toolbox can be used in academic disciplines besides psychology. Because first order logic is such a general language, the material in this paper seems applicable there as well. A second area of development is to complement existing predicates in Theory Toolbox with additional ones. For example, a predicate that checks whether a theory subsumes another theory, a predicate that does better numerical optimization, and a predicate that computes a better measure of falsifiability (such as some measure of entropy), and so on. A third goal is to make Theory Toolbox more accessible on the web.[19] At present, it requires a download and a SWI Prolog installation. Ideally, instead, users should be able to go to a web page, upload a theory or provide a link to it, and run the predicates in Theory Toolbox from this page.

The theory examples in the paper can also be found in the GitHub repository for Theory Toolbox. GitHub is an online software hosting and version control platform. Generally speaking, we think that GitHub, or some other version control system, is an ideal place for storing theories in psychology, for three reasons: (1) Different research teams can collaborate online around a theory; (2) It is possible to trace the history of a theory with a detailed view of changes; (3) The theory can be cloned to a local computer in order to run queries on it (e.g. with Theory Toolbox). In this way all involved researchers can instantly update and instantly query the latest version of the theory on which they collaborate. And if the repository is public, the latest version of the theory is instantly available to the general public.

[18] More information about tabling can be found in this link: https://www.swi-prolog.org/pldoc/man?section=tabling.

[19] In fact, there is an online version of SWI Prolog, called Swish, which can run Prolog code in a web browser. The problem, however, is that Theory Toolbox, like any other meta-interpreter, needs to be able to call arbitrary goals, and such calls are blocked by Swish for security reasons (which is understandable). A possible solution might be to implement a custom web server written in SWI Prolog that lifts these limitations.

5. Conclusions

In summary, we hope that we have succeeded in attracting readers' interest in LP even if this is a complex topic. We find it fascinating that researchers in psychology have the opportunity to collaborate online to build expressive and precise theories in first order logic, and that such theories can be handed to algorithms that conduct scientific inference. Accuracy is an important sub-goal in the route towards the more general goal of using science to improve human life.

CRediT authorship contribution statement

Jean-Christophe Rohner: Conceptualization, Methodology, Software, Validation, Writing - original draft, Writing - review & editing, Visualization, Project administration. **Håkan Kjellerstrand:** Conceptualization, Methodology, Software, Validation, Writing - review & editing.

References

Adam, C., Herzig, A., & Longin, D. (2009). A logical formalization of the OCC theory of emotions. *Synthese, 168*(2), 201–248.

Ajzen, I. (1991). The theory of planned behavior. *Organizational Behavior and Human Decision Processes, 50*(2), 179–211.

American Psychiatric Association. (2013). *Diagnostic and statistical manual of mental disorders (DSM-5®)*. American Psychiatric Association Publishing.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review, 111*(4), 1036–1060.

Baddeley, A. (2012). Working memory: Theories, models, and controversies. *Annual Review of Psychology, 63*, 1–29.

Bandura, A. (2004). Observational learning. In J. H. Byrne (Ed.), *Learning and memory (2nd ed. ed.* (pp. 482–484). Macmillan Reference.

Baron-Cohen, S. J. (2000). *Understanding other minds: Perspectives from developmental cognitive neuroscience*. Oxford University Press.

Batchelder, W. H., & Riefer, D. M. (1999). Theoretical and empirical review of multinomial process tree modeling. *Psychonomic Bulletin & Review, 6*(1), 57–86.

Bielza, C., & Larrañaga, P. J. (2014). Bayesian networks in neuroscience: A survey. *Frontiers in Computational Neuroscience, 8*, 1–23, 131.

Bond, A. H. (1999). Describing behavioral states using a system model of the primate brain. *American Journal of Primatology, 49*(4), 315–338.

Bratko, I. (2001). *Prolog programming for artificial intelligence*. Pearson education.

Bunnin, N., & Yu, J. (2008). *The Blackwell dictionary of western philosophy*. John Wiley & Sons.

Clocksin, W. F., & Mellish, C. S. (2012). *Programming in prolog: Using the ISO standard*. Springer Science & Business Media.

Cohen, P., West, S. G., & Aiken, L. S. (2014). *Applied multiple regression/correlation analysis for the behavioral sciences*. Psychology Press.

Colmerauer, A., & Roussel, P. (1996). The birth of Prolog. In T. J. Bergin, & R. G. Gibson (Eds.), *History of programming languages* (pp. 331–367). ACM Press and Addison Wesley.

Crookes, D. (1988). Using Prolog to present abstract machines. *ACM SIGCSE Bulletin, 20*(3), 8–12.

De Raedt, L., Kimmig, A., & Toivonen, H. (2007). ProbLog: A probabilistic prolog and its application in link discovery. In *International joint conferences on artificial intelligence* (pp. 2468–2473) (Hyderabad).

Fishbein, M., & Ajzen, I. (1974). Attitudes towards objects as predictors of single and multiple behavioral criteria. *Psychological Review, 81*(1), 59.

Flax, L. (2007). Cognitive modelling applied to aspects of schizophrenia and autonomic computing. *International Journal of Cognitive Informatics and Natural Intelligence, 1*(2), 58–72.

Fuchs, N. E., Kaljurand, K., & Kuhn, T. (2008). Attempto controlled English for knowledge representation. In C. Baroglio, P. A. Bonatti, J. Małuszyński, M. Marchiori, A. Polleres, & S. Schaffert (Eds.), *Reasoning web: 4th international summer school 2008, Venice, Italy, september 7-11, 2008, tutorial lectures* (pp. 104–124). Springer Berlin Heidelberg.

Gavrilova, T., & Andreeva, T. (2012). Knowledge elicitation techniques in a knowledge management context. *Journal of Knowledge Management, 16*(4), 523–537.

Gordon, A. S., & Hobbs, J. R. (2017). *A formal theory of commonsense psychology: How people think people think*. Cambridge University Press.

Griffiths, T. L., & Tenenbaum, J. B. (2007). *Two proposals for causal grammars. Causal learning: Psychology, philosophy, and computation* (pp. 323–345).

Holyoak, K. J., & Morrison, R. G. (2005). *The Cambridge handbook of thinking and reasoning (137)*. Cambridge University Press.

Holzbaur, C. (1995). *OFAI clp (q,r) manual, edition 1.3.3. Austrian research Institute for artificial intelligence*.

Jaynes, E. T. (2003). *Probability theory: The logic of science*. Cambridge university press.

Jobin, A., Ienca, M., & Vayena, E. (2019). The global landscape of AI ethics guidelines. *Nature Machine Intelligence, 1*(9), 389–399.

Koslowski, B. (2013). *Scientific reasoning: Explanation, confirmation bias, and scientific practice*. Springer Publishing Company.

Kriegeskorte, N., & Douglas, P. K. (2018). Cognitive computational neuroscience. *Nature Neuroscience, 21*(9), 1148–1160.

Lally, A., & Fodor, P. (2011). *Natural language processing with Prolog in the IBM Watson system*. https://www.cs.nmsu.edu/ALP/2011/03/natural-language-processing-with-prolog-in-the-ibm-watson-system/.

Lazarus, R. S. (1991). Progress on a cognitive-motivational-relational theory of emotion. *American Psychologist, 46*(8), 819–834.

Mackie, J. L. (1974). *The cement of the universe: A study of causation*. Clarendon Press.

Marwala, T. (2019). *Handbook of machine learning*. World Scientific.

McArdle, J. J., & Kadlec, K. M. (2013). Structural equation models. In *The Oxford handbook of quantitative methods* (Vol. 2, pp. 295–338). Oxford University Press.

Merritt, D. (1989). *Building expert systems in Prolog*. New York: Springer Verlag.

Murdoch, W. J., Singh, C., Kumbier, K., Abbasi-Asl, R., & Yu, B. (2019). Definitions, methods, and applications in interpretable machine learning. In *Proceedings of the National Academy of Sciences, 116* pp. 22,071–22080), 44.

Nilsson, U., & Maluszynski, J. (1995). *Logic, programming and prolog (2ed)*. John Wiley & Sons.

Nutt, D., King, L., Saulsbury, W., & Blakemore, C. (2007). Development of a rational scale to assess the harm of drugs of potential misuse. *Lancet, 369*, 1047–1053.

Pearl, J. (2009). Causal inference in statistics: An overview. *Statistics Surveys, 3*, 96–146.

Popper, K. R. (1972). *The logic of scientific discovery*. Hutchinson.

Pylyshyn, Z. (1973). What the mind's eye tells the mind's brain: A critique of mental imagery. *Psychological Bulletin, 80*(1), 1–24.

Rayner, M., Hockey, B. A., Renders, J.-M., Chatzichrisafis, N., & Farrell, K. (2005). Spoken language processing in the Clarissa procedure browser. *Natural Language Engineering, 1*(1), 1–28.

Riguzzi, F. (2018). *Foundations of probabilistic logic programming*. River Publishers.

Robinson, J. A. (1965). A machine-oriented logic based on the resolution principle. *Journal of the ACM, 12*(1), 23–41.

Rohner, J. C. (2020). *Theory toolbox. Retrieved 15 sep 2020 from.* https://github.com/JeanChristopheRohner/theory-toolbox.

Russell, S. J., Norvig, P., & Davis, E. (2010). *Artificial intelligence : A modern approach* (3rd ed.). Prentice Hall.

Shadish, W. R., Cook, T. D., & Campbell, D. T. (2002). *Experimental and quasi-experimental designs for generalized causal inference. Houghton*. Mifflin and Company.

Shapiro, S., & Kouri Kissel, T. (2018). Classical logic. In E. Zalta (Ed.), *Stanford encyclopedia of philosophy. Metaphysics research lab*. Stanford University.

Shapiro, E., & Sterling, L. (1986). *The art of Prolog*. The Massachusetts Institute of Technology.

Shaughnessy, J. J., Zechmeister, E. B., & Zechmeister, J. S. (2000). *Research methods in psychology*. McGraw-Hill.

Skinner, B. F. (1953). *Science and human behavior*. Simon and Schuster.

Smith, C. A., & Ellsworth, P. C. (1985). Patterns of cognitive appraisal in emotion. *Journal of Personality and Social Psychology, 48*(4), 813.

Smith, C. A., & Lazarus, R. S. (1993). Appraisal components, core relational themes, and the emotions. *Cognition & Emotion, 7*(3–4), 233–269.

Srinivasan, A., Muggleton, S. H., Sternberg, M. J. E., & King, R. D. (1996). Theories for mutagenicity: A study in first-order and feature-based induction. *Artificial Intelligence, 85*(1), 277–299.

Stanford, K. (2017). Underdetermination of scientific theory. In E. N. Zalta (Ed.), *The stanford encyclopedia of philosophy. Metaphysics research lab*. Stanford University.

Sterling, L., Bundy, A., Byrd, L., O'Keefe, R., & Silver, B. (1989). Solving symbolic equations with PRESS. *Journal of Symbolic Computation, 7*(1), 71–84.

Sterling, L., & Shapiro, E. Y. (1994). *The art of Prolog: Advanced programming techniques*. Massachusetts Institute of Technology Press.

Triska, M. (2020). The power of prolog. Retrieved 9 May 2020 from https://www.metalevel.at/prolog.

Wielemaker, J., Schrijvers, T., Triska, M., & Lager, T. (2012). *SWI-prolog. Theory and practice of logic programming* (Vol. 12, pp. 67–96), 1-2.

Wilson, W. (2005). Use of logic programming for complex business rules. In *International conference on logic programming* (pp. 14–20). Springer.