

# An Algebraic Approach to Stochastic ASP

Francisco Coelho<sup>1,2</sup>, Bruno Dinis<sup>1,3</sup>, Dietmar Seipel<sup>4</sup>, Salvador Abreu<sup>1,2</sup>

<sup>1</sup>University of Évora <sup>2</sup>NOVA LINES <sup>3</sup>CIMA <sup>4</sup>Universität Würzburg

{fc, bruno.dinis, spa}@uevora.pt dietmar.seipel@uni-wuerzburg.de

## Abstract

We address the problem of extending probability from the total choices of an ASP program to the stable models, and from there to general events. **lengthen abstract** Our approach is algebraic in the sense that it relies on an equivalence relation over the set of events and uncertainty is expressed with variables and polynomial expressions. We illustrate our methods with two examples, one of which shows a connection to bayesian networks.

## 1 Introduction and Motivation

A major limitation of logical representations in real world applications is the implicit assumption that the background knowledge is perfect. This assumption is problematic if data is noisy, which is often the case. Here we aim to explore how answer set program programs with probabilistic facts can lead to characterizations of probability functions on the program’s domain, which is not straightforward in the context of answer set program, as explained below (see also (Verreet et al. 2022; Pajunen and Janhunen 2021; Cozman and Mauá 2020; Baral, Gelfond, and Rushton 2009)). Unlike current systems such as ProbLog (De Raedt et al. 2007), P-log (Baral, Gelfond, and Rushton 2009) or LP<sup>MLN</sup> (Lee and Wang 2016), that derive a probability distribution from a program, in our system some choices are represented by a parameter that can be later estimated from further information, *e.g.* observations. This approach enables later refinement and scoring of a partial program of a model from additional evidence.

Answer set program (ASP) (Lifschitz 2002; Lifschitz 2008) is a logic programming paradigm based on the stable model (SM) semantics of normal programs (NPs). ASP programs represent a problem and the resulting models (*answer sets*) can be found using the latest advances in SAT solving technology (Gebser et al. 2011; Adrian et al. 2018; Niemelä and Simons 1997) or through top-down searching (Alberti et al. 2017; Arias et al. 2020; Marple, Salazar, and Gupta 2017). Unlike Prolog, ASP is a truly declarative language that supports language constructs such as disjunction in the head of a rule, choice rules, and both hard and weak constraints. The distribution semantics (DS) (Sato 1995; Riguzzi 2022) is a key approach to extend logical representations with probabilistic reasoning.

We can foresee two key applications of this extended probability:

1. Support probabilistic reasoning/tasks on the program domain, *i.e.* the set of all events,  $\mathcal{E}$ .
2. Given a dataset and a divergence measure, the program can be scored (by the divergence w.r.t. the *empiric* distribution of the dataset), and weighted or sorted amongst other programs. These are key ingredients in algorithms searching, for example, optimal models of a dataset.

■ **Stress propagation of probabilities.** To extend probabilities from total choices we start with the stance that *a program describes an observable system, that the stable models are all the possible states of that system and that observations (i.e. events) are stochastic* — one observation can be sub-complete (a proper subset of a SM) or super-complete (a proper superset of a SM), and might not determine the real state of the system. From here, probabilities must be extended from total choices (TCs) to SMs and then to any event. ■ **State that we are propagating from the models, not the programs.** This extension process starts with a critical problem, illustrated by the example in section 3, concerning situations where multiple SMs, *ab* and *ac*, result from a single TC, *a*, but there is not enough information (in the program) to assign a single probability to each SM. We propose to address this issue by using algebraic variables to describe that lack of information and then estimate the value of those variables from empirical data. This lack of uniqueness is also addressed in (Cozman and Mauá 2020) along a different approach, using credal sets.

In another related work (Verreet et al. 2022) epistemic uncertainty (or model uncertainty) is considered as a lack of knowledge about the underlying model, that may be mitigated via further observations. This seems to presuppose a bayesian approach to imperfect knowledge in the sense that having further observations allows to improve/correct the model. Indeed, that approach uses Beta distributions on the total choices in order to be able to learn a distribution on the events. This approach seems to be specially fitted to being able to tell when some probability lies beneath some given value. Our approach seems to be similar in spirit, while remaining algebraic in the way that the extension of probabilities is addressed.

The example in section 3 uses the code available in the

project’s repository<sup>1</sup>, developed with the *Julia* programming language (Bezanson et al. 2017), and the *Symbolics* (Gowda et al. 2021) libraries.

## 2 Syntax and Semantics of Stochastic ASP

■ Underline that the scope of this syntax is just to start studying our propagation method.

We start this overview with the syntax and ASP semantics of (propositional) normal programs and then proceed to the case of probabilistic annotations.

Let  $\mathcal{A}$  be a finite set. A *positive literal*, or *atom*, is any  $a \in \mathcal{A}$  and a *negative literal* is  $\neg a$ , also denoted  $\bar{a}$ , for  $a \in \mathcal{A}$ . A *literal* is a positive or a negative literal. A *fact* is a literal. For literals  $x_1, \dots, x_n$ , a *conjunction* is  $x_1 \wedge \dots \wedge x_n$  and a *disjunction* is  $x_1 \vee \dots \vee x_n$ . A *normal rule* is  $h \leftarrow b$  where  $h$  is a literal, the *head*, and  $b$  a conjunction, the *body*. In a *choice rule* the head is a disjunction. An *answer set program* (ASP) is a set of facts and (both normal and choice) rules, denoted, resp.  $\mathcal{F}(P)$ ,  $\mathcal{R}(P)$ , or simply  $\mathcal{F}$ ,  $\mathcal{R}$ . ■ Notice that choice rules can be converted to a normal rules (Gebser et al. 2022).

The semantics of an ASP can have different definitions (Lifschitz 2008). A common definition is as follows (Gelfond and Lifschitz 1988). Let  $P$  be a normal program. The *reduct* of  $P$  relative to the set of atoms  $X$  results from (i) deleting rules that contains a literal  $\neg x$  with  $x \in X$  and then (ii) deleting literals  $\neg y$  from the bodies of the remaining rules. Now,  $\mathcal{M}(P)$ , or simply  $\mathcal{M}$ , is a *stable model* (SM) of  $P$  if it is the minimal model of the reduct of  $P$  relative to  $\mathcal{M}$ .

Another promising approach to handle the generation of stable models is the one supported by *s(CASP)*, a system that can evaluate ASP programs with function symbols (functors) and constraints without grounding them either before or during execution, using a method similar to SLD resolution (Marple, Salazar, and Gupta 2017; Arias et al. 2020). This enables the generation of human readable explanations of the results of programs and addresses two major issues of grounding-based solvers, that 1) either do not support function symbols or, using finite domains, lead to exponential groundings of a program and 2) compute the complete model of the grounded program when, in some scenarios, it is desirable to compute only a partial stable model containing a query.

*Stochastic Answer Set Programs* (SASPs) extend ASP by adding facts with probabilistic annotations: A *probabilistic fact* (PF) is  $a : p$  where  $a$  is an atom and  $p \in [0, 1]$ . We denote the set of probabilistic facts of a program by  $\mathcal{P}$ , and  $\mathcal{A}_{\mathcal{P}}$  the set of positive literals in  $\mathcal{P}$ . ■ Relate to (Kifer and Subrahmanian 1992) and state that our propagation of probability is through the models of the program, not its syntax.

The *derived program* of a SASP is obtained by replacing each probabilistic fact  $a : p$  by  $a \vee \bar{a}$ . The *stable models* of a SASP program are the stable models of its derived program. The set of SMs of a (derived or) SASP program is denoted  $\mathcal{M}$ .

An *event* is a set of literals. We denote the set of events by  $\mathcal{E}$ . For  $e \in \mathcal{E}$ , if there exists a positive literal  $x$  such that  $\{x, \bar{x}\} \subseteq e$  then  $e$  is *inconsistent*; otherwise  $e$  is *consistent*. The set of consistent events is denoted by  $\mathcal{W}$ .

A disjunctive head  $a \vee \bar{a}$  in the derived program represents a single *choice*, either  $a$  or  $\bar{a}$ . A *total choice* of the derived program, and of the SASP program, is  $t = \{a' \mid a : p \in \mathcal{P}\}$  where each  $a'$  is either  $a$  or  $\bar{a}$ . We denote  $\mathcal{T}$  the set of total choices of a SASP or derived program. The *probability of the total choice*  $t \in \mathcal{T}$  is

$$P(T = t) = \prod_{\substack{a:p \in \mathcal{P}, \\ a \in t}} p \prod_{\substack{a:p \in \mathcal{P}, \\ \bar{a} \in t}} \bar{p} \quad (1)$$

where  $T$  is a random variable whose values are total choices.

Some stable models are entailed from some total choices while other SMs are entailed by other TCs. We write  $\mathcal{M}(t)$  to represent the set of SMs entailed by  $t \in \mathcal{T}$ .

**BEGIN other approaches and systems.**

A related approach to define semantics of normal programs with probabilistic facts is described in (Cozman and Mauá 2020), where  $P(T = t)$  is defined like in eq. (1) but then, for  $a \in \mathcal{A}$ ,  $P(a \mid t)$  is unknown but bounded by  $\underline{P}(a \mid t)$  and  $\bar{P}(a \mid t)$ , that can be explicitly estimated from the program.

- P-Log still missing.
- draft: Problog2: (Fierens et al. 2015; Verreet et al. 2022).
  - A Problog program specifies a probability distribution over possible worlds.
  - world is a model of  $C \cup R$  where  $C$  is a total choice and  $R$  the set of rules of a program.
  - Semantics only defined for *sound* programmas (Riguzzi and Swift 2013) i.e., programs for which each possible total choice  $C$  leads to a well-founded model that is two-valued or ‘total’ (Riguzzi and Swift 2013; Van Gelder et al. 1991). A sufficient condition for this is that the rules in the Problog program are locally stratified (Van Gelder et al. 1991). In particular, this trivially holds for all negation-free programs.
  - Probability of a possible world that is a model of the program is the probability of the total choice. Otherwise probability is 0.
- draft: LP<sup>MLN</sup>: (Lee and Wang 2016; Lee and Yang 2017)
  - Extends Problog
  - based on Markov Logic (Richardson and Domingos 2006)
  - weighted rules  $a \leftarrow b, n$  where  $a$  is disjunction of atoms,  $b$  is conjunction of atoms,  $n$  is constructed from atoms using conjunction, disjunction and negation.
  - For each model there is a unique maximal set of rules that are satisfied by it. The respective weights determine the probability of the model.

**END other approaches and systems.**

<sup>1</sup><https://git.xdi.uevora.pt/fc/sasp>

■ Our goal is to extend the probability  $P(T = t)$  in eq. (1) (which is, indeed, a product of Bernoulli distributions (Teugels 1990)), from total choices to the program’s events,  $P(E = e)$ .

### 3 A Simple but Fruitful Example

In this section we consider a somewhat simple example that showcases the problem of extending probabilities from total choices to stable models and then to events. As mentioned before, the main issue arises from the lack of information in the program to assign a single probability to each stable model. This becomes a crucial problem in situations where multiple stable models result from a single total choice. We will come back to this example in appendix A.1, after we present our proposal for extending probabilities from total choices to stable models in section 4.

**Example 1.** Consider the program

$$\begin{aligned} a &: 0.3, \\ b \vee c &\leftarrow a. \end{aligned} \quad (2)$$

The stable models of this SASP are the ones from its derived program:  $\bar{a}, ab$  and  $ac$ . We use short-hand expressions like  $ab$  to denote e.g. the set  $\{a, b\}$ . While it is straightforward to assume  $P(\bar{a}) = 0.7$ , there is no obvious explicit way to assign values to  $P(ab)$  and  $P(ac)$ . Instead, we can use a parameter  $\theta$  as in

$$\begin{aligned} P(ab) &= 0.3\theta, \\ P(ac) &= 0.3(1 - \theta) \end{aligned}$$

to express our knowledge that  $ab, ac$  are events related in a certain way and, simultaneously, our uncertainty about that relation. The parameter  $\theta = \theta_{s,t}$  depends on both the stable model  $s$  and the total choice  $t$ . This uncertainty can then be addressed with the help of adequate distributions, such as empirical distributions from datasets.

If an ASP program is intended to describe some system then:

1. With a probability set for the stable models, we want to extend it to all the events of the program domain.
2. In the case where some statistical knowledge is available, for example, in the form of a distribution, we consider it as “external” knowledge about the parameters, that doesn’t affect the extension procedure described below.
3. Statistical knowledge can be used to estimate parameters and to “score” the program.
4. If that program is only one of many possible candidates then that score can be used, e.g. as fitness, by algorithms searching (optimal) programs of a dataset of observations.
5. If observations are not consistent with the program, then we ought to conclude that the program is wrong and must be changed accordingly.

Currently, we are addressing the problem of extending a probability function (possibly using parameters such as  $\theta$  above), defined on the SMs of a program, to all the events of that program. This extension must satisfy the Kolmogorov

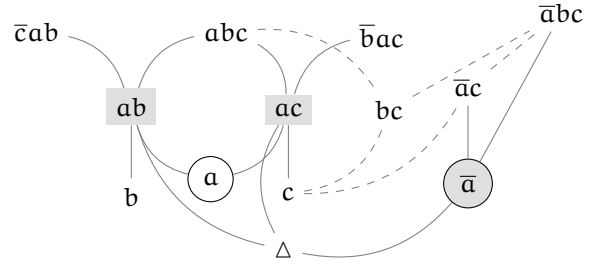


Figure 1: Some events related to the stable models of ex. 1. The circle nodes are total choices and shaded nodes are stable models. Solid lines represent relations with the SMs and dashed lines some sub-set relations with other events. The set of events contained in all stable models, denoted by  $\Delta$ , is empty in this example. Consider denoting the event  $\emptyset$  by  $\lambda$  and using  $\wedge$  instead of  $\Delta$ . We still need to denote the set of events related with all the SMs, but we are being less confusing with this notation.

axioms of probability so that probabilistic reasoning is consistent with the ASP program and follow our interpretation of stable models as the states of an observable system.

As sets, the SMs can have non-empty intersection. But, as states of a system, we assume that SMs are disjoint events, in the following sense:

**Assumption 1.** Stable models are disjoint events: For any set  $X$  of stable models,

$$P(X) = \sum_{s \in X} P(s) \quad (3)$$

Consider the stable models  $ab, ac$  from ex. 1, that result from the rule  $b \vee c \leftarrow a$  and the total choice  $\{a\}$ . Since we intend to associate each stable model with a state of the system,  $ab$  and  $ac$  should be *disjoint* events. So  $b \vee c$  is interpreted as an *exclusive disjunction* and, from that particular rule, no further relation between  $b$  and  $c$  is assumed. This does not prevent that other rules may be added that entail further dependencies between  $b$  and  $c$ , which in turn may change the stable models.

By not making distribution assumptions on the rules of the program we can state such properties on the semantics of the program, as we’ve done in assumption 1.

## 4 Extending Probabilities

The diagram in fig. 1 illustrates the problem of extending probabilities from total choices to stable models and then to general events in an *edge-wise* process, where the value in a node is defined from the values in its neighbors. This quickly leads to coherence problems concerning probability, with no clear systematic approach. Notice that  $bc$  is not directly related with any stable model therefore propagating values through edges would assign a hard to justify ( $\neq 0$ ) value to  $bc$ . Instead, we propose to base the extension in the relation an event has with the stable models.

### 4.1 An Equivalence Relation

■ Should we bring back the parts moved to the appendix?

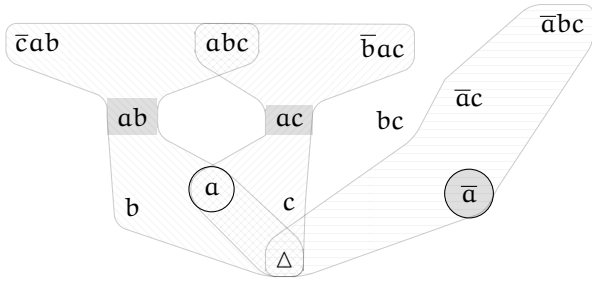


Figure 2: Classes of (consistent) events related to the stable models of ex. 1 are defined through intersections and inclusions. In this picture we can see, for example, the classes  $\{\bar{c}ab, ab, b\}$  and  $\{a, abc\}$ . Different fillings correspond to different classes and, as before, the circle nodes are total choices and shaded nodes are stable models. Notice that  $bc$  is not in a shaded area.

**BEGIN** Draft zone to integrate with (Kifer and Subrahmanian 1992)

We need to define a lattice  $\mathbb{T}$  of “truth values” to interpret any event as an ideal. And do so in a way compatible with (Kifer and Subrahmanian 1992).

And families  $\mathbb{F}_i : \mathbb{T}^i \rightarrow \mathbb{T}$  of total continuous computable functions to define term-annotated clause heads *Maybe* we don’t need this now; we are considering only c-annotations.

**END** Draft zone to integrate (Kifer and Subrahmanian 1992)

Our path to extend probabilities starts with a perspective of stable models as playing a role similar to *prime factors or principal ideals*. The stable models of a program are the irreducible events entailed from that program and any event must be considered under its relation with the stable models.

From ex. 1 and fig. 2 consider the SMs  $\bar{a}, ab, ac$  and events  $a, abc$  and  $c$ . While  $a$  is related with (contained in) both  $ab, ac$ , the event  $c$  is related only with  $ac$ . So,  $a$  and  $c$  are related with different SMs. On the other hand,  $abc$  contains with both  $ab, ac$ . So  $a$  and  $abc$  are related with the same stable models.

**Definition 1.** The stable core (SC) of the event  $e \in \mathcal{E}$  is

$$[e] := \{s \in \mathcal{M} \mid s \subseteq e \vee e \subseteq s\}. \quad (4)$$

We now define an equivalence relation so that two events are related if either both are inconsistent or both are consistent and, in the latter case, with the same stable core.

**Definition 2.** For a given program, let  $u, v \in \mathcal{E}$ . The equivalence relation  $\sim$  is defined by

$$u \sim v : \Leftrightarrow u, v \notin \mathcal{W} \vee (u, v \in \mathcal{W} \wedge [u] = [v]). \quad (5)$$

Observe that the minimality of stable models implies that, in definition 1, either  $e$  is a stable model or at least one of  $\exists s (s \subseteq e), \exists s (e \subseteq s)$  is false. This equivalence relation defines a partition on the set of events, where each class holds a unique relation with the stable models. In particular we denote each class by

$$[e]_{\sim} = \begin{cases} \perp := \mathcal{E} \setminus \mathcal{W} & \text{if } e \in \mathcal{E} \setminus \mathcal{W}, \\ \{u \in \mathcal{W} \mid [u] = [e]\} & \text{if } e \in \mathcal{W}. \end{cases} \quad (6)$$

The combinations of the stable models, together with the set of inconsistent events  $\perp$ , form a set of representatives. Consider again ex. 1. As previously mentioned, the stable models are the elements of  $\mathcal{M} = \{\bar{a}, ab, ac\}$  so the quotient set of this relation is

$$[\mathcal{E}]_{\sim} = \left\{ \begin{array}{ccc} \perp, & \diamond, & [\bar{a}]_{\sim}, \\ [ab]_{\sim}, & [ac]_{\sim}, & [\bar{a}, ab]_{\sim}, \\ [\bar{a}, ac]_{\sim}, & [ab, ac]_{\sim}, & [\bar{a}, ab, ac]_{\sim} \end{array} \right\}, \quad (7)$$

where  $\diamond$  denotes, with abuse of notation, both the class of *independent* events  $e$  such that  $[e] = \emptyset$  and its core, while  $\Delta = \bigcap_{s \in \mathcal{M}} s$ , is the set of events contained in all SMs. We have:

$[e]$	$[e]_{\sim}$	$\#[e]_{\sim}$
$\perp$	$\bar{a}a, \dots$	37
$\diamond$	$\bar{b}, \bar{c}, bc, \bar{b}a, \bar{b}c, \bar{b}\bar{c}, \bar{c}a, \bar{c}b, \bar{b}\bar{c}a$	9
$\bar{a}$	$\bar{a}, \bar{a}b, \bar{a}c, \bar{a}\bar{b}, \bar{a}\bar{c}, \bar{a}bc, \bar{a}\bar{c}b, \bar{a}bc, \bar{a}\bar{b}c$	9
$ab$	$b, ab, \bar{c}ab$	3
$ac$	$c, ac, bac$	3
$\bar{a}, ab$	$\emptyset$	0
$\bar{a}, ac$	$\emptyset$	0
$ab, ac$	$a, abc$	2
$\bar{a}, ab, ac$	$\Delta$	1
$[\mathcal{E}]_{\sim}$	$\mathcal{E}$	64

Since all events within an equivalence class are in relation with a specific set of stable models, *measures, including probability, should be constant within classes*:

$$\forall u \in [e]_{\sim} (\mu(u) = \mu(e)).$$

In general, we have *much more* stable models than literals but their combinations are still *much less* than events. Nevertheless, the equivalence classes allow us to propagate probabilities from total choices to events, as explained in the next subsection.

In this specific case, instead of dealing with  $64 = 2^6$  events, we consider only the  $9 = 2^3 + 1$  classes, well defined in terms of combinations of the stable models.

## 4.2 From Total Choices to Events

Our path to set a distribution on  $\mathcal{E}$  starts with the more general problem of extending *measures*, since extending *probabilities* easily follows by means of a suitable normalization (done in eqs. (15) and (16)), and has two phases:

1. Extension of the probabilities, *as measures*, from the total choices to events.
2. Normalization of the measures on events, recovering a probability.

The “extension” phase, traced by eq. (1) and eqs. (8) to (14), starts with the measure (probability) of total choices,  $\mu(t) = P(T = t)$ , expands it to stable models,  $\mu(s)$ , and then, within the equivalence relation from eq. (5), to (general) events,  $\mu(e)$ , including (consistent) worlds.

**Total Choices.** Using eq. (1), this case is given by

$$\mu_{TC}(t) := P(T = t) = \prod_{c:p < \mathcal{P}, c \in t} p \prod_{c:p \in \mathcal{P}, \bar{c} \in t} \bar{p}. \quad (8)$$



**Stable Models.** Recall that each total choice  $t$ , together with the rules and the other facts of a program, defines the set  $\mathcal{M}(t)$  of stable models associated with that choice. Given a total choice  $t$ , a stable model  $s$ , and variables or values  $\theta_{s,t} \in [0, 1]$  such that  $\sum_{s \in \mathcal{M}(t)} \theta_{s,t} = 1$ , we define

$$\mu(s, t) := \begin{cases} \theta_{s,t} & \text{if } s \in \mathcal{M}(t) \\ 0 & \text{otherwise.} \end{cases} \quad (9)$$

**Classes.** Each class is either the inconsistent class,  $\perp$ , or is represented by some set of stable models.

**Inconsistent Class.** The inconsistent class contains events that are logically inconsistent, thus should never be observed and have measure zero:

$$\mu(\perp, t) := 0.^2 \quad (10)$$

**Independent Class.** An event unrelated with any stable model corresponds to a non-state, according to the program. So the respective measure is also set to zero:

$$\mu(\diamond, t) := 0. \quad (11)$$

**Other Classes.** The extension must be constant within a class, its value should result from the elements in the stable core, and respect<sup>1</sup> assumption 1 (stable models are disjoint):

$$\mu([e]_{\sim}, t) := \mu(\llbracket e \rrbracket, t) = \sum_{s \in \llbracket e \rrbracket} \mu(s, t) \quad (12)$$

and

$$\mu([e]_{\sim}) := \sum_{t \in \mathcal{T}} \mu([e]_{\sim}, t) \mu_{TC}(t). \quad (13)$$

**Events.** Each (general) event  $e$  is in the class defined by its stable core,  $\llbracket e \rrbracket$ . So, denoting by  $\#X$  the number of elements in  $X$ , we set:

$$\mu(e, t) := \begin{cases} \frac{\mu([e]_{\sim}, t)}{\#[e]_{\sim}} & \text{if } \#[e]_{\sim} > 0, \\ 0 & \text{otherwise.} \end{cases} \quad (14)$$

and

$$\mu(e) := \sum_{t \in \mathcal{T}} \mu(e, t) \mu_{TC}(t). \quad (15)$$

The  $\theta_{s,t}$  parameters in eq. (9) express the *program's* lack of information about the measure assignment, when a single total choice entails more than one stable model. In that case, how to distribute the respective measures? Our proposal to address this problem consists in assigning a parameter,  $\theta_{s,t}$ , conditional on the total choice,  $t$ , to each stable model  $s$ . This approach allows the expression of an unknown quantity and future estimation, given more information, *e.g.* observed data.

Equation (12) results from asp. 1 and states that the measure of a class  $[e]_{\sim}$  is the sum over its stable core,  $\llbracket e \rrbracket$ , and eq. (13) *marginalizes* the TCs on eq. (12).

The *normalizing factor* is:

<sup>2</sup>This measure being zero is independent of the total choice.

$$Z := \sum_{e \in \mathcal{E}} \mu(e) = \sum_{[e]_{\sim} \in [\mathcal{E}]_{\sim}} \mu([e]_{\sim}),$$

and now eq. (15) provides a straightforward way to define the *probability of (observation of) a single event*:

$$P(E = e) := \frac{\mu(e)}{Z}. \quad (16)$$

Equation (15) together with external statistical knowledge, can be used to learn about the *initial* probabilities of the literals, that should not (and by prop. 1 can't) be confused with the explicit  $\mu_{TC}$  set in the program.

It is now straightforward to check that  $P(E)$  satisfies the Kolmogorov axioms of probability for  $\Omega = \mathcal{E}$ .

Since total choices are also events, one can ask, for an arbitrary total choices  $t$ , if  $P(T = t) = P(E = t)$  or, equivalently, if  $\mu_{TC}(t) = \mu(t)$ . However, it is easy to see that, in general, that cannot be true. While the domain of the random variable  $T$  is the set of total choices, for  $E$  the domain is much larger, including all the events. Except for trivial programs, where the SMs are the TCs, some events other than total choices have non-zero probability.

**Proposition 1.** *In a program with a stable model that is not a total choice there is at least one  $t \in \mathcal{T}$  such that:*

$$P(T = t) \neq P(E = t). \quad (17)$$

*Proof.* Suppose towards a contradiction that  $P(T = t) = P(E = t)$  for all  $t \in \mathcal{T}$ . Then

$$\sum_{t \in \mathcal{T}} P(E = t) = \sum_{t \in \mathcal{T}} P(T = t) = 1.$$

Hence  $P(E = x) = 0$  for all  $x \in \mathcal{E} \setminus \mathcal{T}$ , in contradiction with the fact that for at least one  $s \in \mathcal{P} \setminus \mathcal{T}$  one has  $P(E = s) > 0$ .  $\square$

The essential conclusion of prop. 1 is that we are dealing with *two distributions*: one, on the TCs, explicit in the annotations of the programs and another one, on the events, and entailed by the explicit annotations *and the structure of the stable models*.

■ To generalize to Bayesian networks, we might use (Cozman and Mauá 2020; Raedt et al. 2016) and (Kießling, Thöne, and Güntzer 1992; Thöne, Güntzer, and Kießling 1997): Any acyclic propositional program can be viewed as the specification of a Bayesian network over binary random variables: the structure of the Bayesian network is the dependency graph; the random variables correspond to the atoms; the probabilities can be read off of the probabilistic facts and rules. Conversely, any Bayesian network over binary variables can be specified by an acyclic nondisjunctive PASP program.

## 5 Discussion and Future Work

This work is a first venture into expressing probability distributions using algebraic expressions derived from a logical program, in particular an ASP. We would like to point out that there is still much to explore concerning the full expressive power of logic programs and ASP programs. So far,

we have not considered recursion, logical variables or functional symbols. Also, there is still little effort to articulate with the related fields, probabilistic logical programming, machine learning, inductive programming, *etc.*

The equivalence relation from definition 2 identifies the  $s \subseteq e$  and  $e \subseteq s$  cases. Relations that distinguish such cases might enable better relations between the models and processes from the stable models.

The example from appendix A.2 shows that the theory, methodology, and tools, from bayesian networks can be adapted to our approach. The connection with Markov Fields (Kindermann and Snell 1980) is left for future work. An example of a “program selection” application (as mentioned in item 4, section 3) is also left for future work.

Related with the remark at the end of appendix A.1, on the tendency of  $\hat{\theta}$  to under- or over- estimate  $\theta$ , notice that the error function in (19) expresses only one of many possible “distances” between the empirical and prior distributions. Variations include normalizing this function by the size of  $\mathcal{E}$  or using the Kullback-Leibler divergence. The key contribution of this function in this work is to find an optimal  $\theta$ . Moreover, further experiments, not included in this paper, with  $\alpha = 0.0$ , lead to  $\hat{\theta} \approx \gamma$ , *i.e.* setting the prior noise to zero leads to full recovering  $\theta$  from the observations.

We decided to set the measure of inconsistent events to 0 but, maybe, in some cases, we shouldn’t. For example, since observations may be affected by noise, one can expect inconsistencies between the literals of an observation to occur.

## Acknowledgements

This work is partly supported by Fundação para a Ciência e Tecnologia (FCT/IP) under contracts UIDB/04516/2020 (NOVA LINC), UIDP/04674/2020 and UIDB/04675/2020 (CIMA).

The authors are grateful to Lúcia Henriques-Rodrigues, Matthias Knorr and Ricardo Gonçalves for valuable comments on a preliminary version of this paper, and Alice Martins for contributions on software development.

## References

Adrian, W. T.; Alviano, M.; Calimeri, F.; Cuteri, B.; Dodaro, C.; Faber, W.; Fuscà, D.; Leone, N.; Manna, M.; Perri, S.; et al. 2018. The asp system dlvs: advancements and applications. *KI-Künstliche Intelligenz* 32:177–179.

Alberti, M.; Bellodi, E.; Cota, G.; Riguzzi, F.; and Zese, R. 2017. cplint on swish: Probabilistic logical inference with a web browser. *Intelligenza Artificiale* 11(1):47–64.

Arias, J.; Carro, M.; Chen, Z.; and Gupta, G. 2020. Justifications for goal-directed constraint answer set programming. *arXiv preprint arXiv:2009.10238*.

Bal, C.; Gelfond, M.; and Rushton, N. 2009. Probabilistic reasoning with Answer Sets. *Theory and Practice of Logic Programming* 9(1):57–144.

Bezanson, J.; Edelman, A.; Karpinski, S.; and Shah, V. B. 2017. Julia: A fresh approach to numerical computing. *SIAM Review* 59(1):65–98.

Cozman, F. G., and Mauá, D. D. 2020. The joy of probabilistic answer set programming: semantics, complexity, expressivity, inference. *International Journal of Approximate Reasoning* 125:218–239.

De Raedt, L.; Kimmig, A.; Toivonen, H.; and Veloso, M. 2007. Prolog: A probabilistic Prolog and its application in link discovery. In *IJCAI 2007, Proceedings of the 20th international joint conference on artificial intelligence*, 2462–2467. IJCAI-INT JOINT CONF ARTIF INTELL.

Fierens, D.; Van den Broeck, G.; Renkens, J.; Shterionov, D.; Gutmann, B.; Thon, I.; Janssens, G.; and De Raedt, L. 2015. Inference and learning in probabilistic logic programs using weighted boolean formulas. *Theory and Practice of Logic Programming* 15(3):358–401.

Gebser, M.; Kaufmann, B.; Kaminski, R.; Ostrowski, M.; Schaub, T.; and Schneider, M. 2011. Potassco: The Potsdam answer set solving collection. *AI Communications* 24(2):107–124.

Gebser, M.; Kaminski, R.; Kaufmann, B.; and Schaub, T. 2022. *Answer set solving in practice*. Springer Nature.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In *ICLP/SLP*, volume 88, 1070–1080. Cambridge, MA.

Geman, S., and Geman, D. 1984. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-6(6):721–741.

Gowda, S.; Ma, Y.; Cheli, A.; Gwozdz, M.; Shah, V. B.; Edelman, A.; and Rackauckas, C. 2021. High-performance symbolic-numerics via multiple dispatch. *arXiv preprint arXiv:2105.03949*.

Kießling, W.; Thöne, H.; and Güntzer, U. 1992. Database support for problematic knowledge. In *Advances in Database Technology - EDBT’92: 3rd International Conference on Extending Database Technology Vienna, Austria, March 23–27, 1992 Proceedings* 3, 421–436. Springer.

Kifer, M., and Subrahmanian, V. 1992. Theory of generalized annotated logic programming and its applications. *The Journal of Logic Programming* 12(4):335–367.

Kindermann, R., and Snell, J. L. 1980. *Markov random fields and their applications*, volume 1 of *Contemporary Mathematics*. American Mathematical Society, Providence, RI.

Lee, J., and Wang, Y. 2016. Weighted rules under the stable model semantics. In *Fifteenth international conference on the principles of knowledge representation and reasoning*.

Lee, J., and Yang, Z. 2017. Lpmln, Weak Constraints, and P-log. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Lifschitz, V. 2002. Answer set programming and plan generation. *Artificial Intelligence* 138(1):39–54.

Lifschitz, V. 2008. Twelve definitions of a stable model. In *International Conference on Logic Programming*, 37–51. Springer.

Marple, K.; Salazar, E.; and Gupta, G. 2017. Computing

stable models of normal logic programs without grounding. *arXiv preprint arXiv:1709.00501*.

Niemelä, I., and Simons, P. 1997. Smodels - an implementation of the stable model and well-founded semantics for normal logic programs. In *Logic Programming And Nonmonotonic Reasoning: 4th International Conference, LPNMR'97 Dagstuhl Castle, Germany, July 28–31, 1997 Proceedings 4*, 420–429. Springer.

Pajunen, J., and Janhunen, T. 2021. Solution enumeration by optimality in Answer Set Programming. *Theory and Practice of Logic Programming* 21(6):750–767.

Pearl, J. 1988. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. The Morgan Kaufmann Series in Representation and Reasoning. Morgan Kaufmann, San Mateo, CA.

Raedt, L. D.; Kersting, K.; Natarajan, S.; and Poole, D. 2016. Statistical relational artificial intelligence: Logic, probability, and computation. *Synthesis lectures on artificial intelligence and machine learning* 10(2):1–189.

Riguzzi, F. 2022. *Foundations of Probabilistic Logic Programming: Languages, Semantics, Inference and Learning*. New York: River Publishers, 1 edition.

Sato, T. 1995. A statistical learning method for logic programs with distribution semantics. In *International Conference on Logic Programming*.

Teugels, J. L. 1990. Some representations of the multivariate Bernoulli and binomial distributions. *J. Multivariate Anal.* 32(2):256–268.

Thöne, H.; Güntzer, U.; and Kießling, W. 1997. Increased robustness of bayesian networks through probability intervals. *International Journal of Approximate Reasoning* 17(1):37–76.

Verreet, V.; Derkinderen, V.; Dos Martires, P. Z.; and De Raedt, L. 2022. Inference and learning with model uncertainty in probabilistic logic programs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 10060–10069.

## A Developed Examples

Here we apply the methods from section 4 to ex. 1 and to a well known bayesian network: the Earthquake, Burglar, Alarm toy problem.

### A.1 The SBF Example

We continue with the program from eq. (2).

**Total choices.** The total choices, and respective stable models, are

Total choice	Stable models	$\mu_{TC}(t)$
$\bar{a}$	ab, ac	0.3
$\bar{a}$	$\bar{a}$	$\overline{0.3} = 0.7$

**Stable models.** The  $\theta_{s,t}$  parameters in this example are

$\theta_{s,t}$	$\bar{a}$	$a$
$\bar{a}$	1	0
ab	0	$\theta$
ac	0	$\bar{\theta}$

with  $\theta \in [0, 1]$ .

**Classes.** Following the definitions in eqs. (4) to (6) and (10) to (12) we get the quotient set (ignoring  $\perp$  and  $\diamond$ ), and measures in table 1.

**Prior Distributions.** Following the above values (in rational form), and considering the inconsistent and independent classes (resp.  $\perp, \diamond$ ) (labeling  $P(E = e)$  by  $P_e$  and  $P(E \in [e]_{\sim})$  by  $P[e]_{\sim}$ )

$\llbracket e \rrbracket$	$\#[e]_{\sim}$	$\mu([e]_{\sim})$	$\mu(e)$	$P_e$	$P[e]_{\sim}$
$\perp$	37	0	0	0	0
$\diamond$	9	0	0	0	0
$\bar{a}$	9	$\frac{7}{10}$	$\frac{7}{90}$	$\frac{7}{207}$	$\frac{7}{23}$
ab	3	$\frac{3}{10}\theta$	$\frac{1}{10}\theta$	$\frac{1}{23}\theta$	$\frac{3}{23}\theta$
ac	3	$\frac{3}{10}\bar{\theta}$	$\frac{1}{10}\bar{\theta}$	$\frac{1}{23}\bar{\theta}$	$\frac{3}{23}\bar{\theta}$
$\bar{a}, ab$	0	$\frac{7+3\theta}{10}$	0	0	0
$\bar{a}, ac$	0	$\frac{7+3\bar{\theta}}{10}$	0	0	0
ab, ac	2	$\frac{3}{10}$	$\frac{3}{20}$	$\frac{3}{46}$	$\frac{3}{23}$
$\bar{a}, ab, ac$	1	1	1	$\frac{10}{23}$	$\frac{10}{23}$
$Z = \frac{23}{10}$					

So the prior distributions, denoted by the random variable  $E$ , of events and classes are listed in table 2.

**Testing the Prior Distributions.** These results can be tested by simulation in a two-step process, where (1) a “system” is simulated, to gather some “observations” and then (2) empirical distributions from those samples are related with the prior distributions from eq. (18). Tables 3 and 4 summarize some of those tests, where datasets of  $n = 1000$  observations are generated and analyzed.

**Simulating a System.** Following some criteria, more or less related to the given program, a set of events, that represent observations, is generated. Possible simulation procedures include:

- *Random.* Each sample is a Random Set of Literals (RSL). Additional sub-criteria may require, for example, consistent events, a Random Consistent Event (RCE) simulation.
- *Model+Noise.* Gibbs’ sampling (Geman and Geman 1984) tries to replicate the program model and also to add some noise. For example, let  $\alpha, \beta, \gamma \in [0, 1]$  be some parameters to control the sample generation. The first parameter,  $\alpha$  is the “out of model” samples ratio;  $\beta$  represents the choice  $a$  or  $\bar{a}$  (explicit in the model) and  $\gamma$  is the simulation representation of  $\theta$ . A single sample is then generated following the probabilistic choices below:

$$\left\{ \begin{array}{l} \alpha \text{ by RCE} \\ \left\{ \begin{array}{l} \beta \bar{a} \\ \gamma \left\{ \begin{array}{l} ab \\ ac \end{array} \right\} \end{array} \right. \end{array} \right.$$

$\llbracket e \rrbracket$	$\mu(s, \bar{a})$			$\mu(s, a)$			$\mu([e]_{\sim}, \bar{a})$	$\mu([e]_{\sim}, a)$	$\mu([e]_{\sim})$
	$\bar{a}$	$ab$	$ac$	$\bar{a}$	$ab$	$ac$	$\mu_{TC} = 0.7$	$\mu_{TC} = 0.3$	
$\bar{a}$	$\boxed{1}$	0	0	$\boxed{0}$	$\theta$	$\bar{\theta}$	1	0	0.7
$ab$	1	$\boxed{0}$	0	0	$\boxed{\theta}$	$\bar{\theta}$	0	$\theta$	$0.3\theta$
$ac$	1	0	$\boxed{0}$	0	$\theta$	$\boxed{\bar{\theta}}$	0	$\bar{\theta}$	$0.3\bar{\theta}$
$\bar{a}, ab$	$\boxed{1}$	$\boxed{0}$	0	$\boxed{0}$	$\boxed{\theta}$	$\bar{\theta}$	1	$\theta$	$0.7 + 0.3\theta$
$\bar{a}, ac$	$\boxed{1}$	0	$\boxed{0}$	$\boxed{0}$	$\theta$	$\boxed{\bar{\theta}}$	1	$\bar{\theta}$	$0.7 + 0.3\bar{\theta}$
$ab, ac$	1	$\boxed{0}$	$\boxed{0}$	0	$\boxed{\theta}$	$\boxed{\bar{\theta}}$	0	$\theta + \bar{\theta} = 1$	0.3
$\bar{a}, ab, ac$	$\boxed{1}$	$\boxed{0}$	$\boxed{0}$	$\boxed{0}$	$\boxed{\theta}$	$\boxed{\bar{\theta}}$	1	$\theta + \bar{\theta} = 1$	1

Table 1: TODO: caption this

$\llbracket e \rrbracket$	$\perp$	$\diamond$	$\bar{a}$	$ab$	$ac$	$\bar{a}, ab$	$\bar{a}, ac$	$ab, ac$	$\bar{a}, ab, ac$
$P(E = e)$	0	0	$\frac{7}{207}$	$\frac{1}{23}\theta$	$\frac{1}{23}\bar{\theta}$	0	0	$\frac{3}{46}$	$\frac{10}{23}$
$P(E \in [e]_{\sim})$	0	0	$\frac{7}{23}$	$\frac{3}{23}\theta$	$\frac{3}{23}\bar{\theta}$	0	0	$\frac{3}{23}$	$\frac{10}{23}$

(18)

Table 2: TODO: caption this

$\llbracket e \rrbracket$	$\# \{S \in [e]_{\sim}\}$	$P(S \in [e]_{\sim})$	$P(E \in [e]_{\sim})$
$\perp$	0	0	0
$\diamond$	24	$\frac{24}{1000}$	0
$\bar{a}$	647	$\frac{647}{1000}$	$\frac{7}{23}$
$ab$	66	$\frac{66}{1000}$	$\frac{3}{23}\theta$
$ac$	231	$\frac{231}{1000}$	$\frac{3}{23}\bar{\theta}$
$\bar{a}, ab$	0	0	0
$\bar{a}, ac$	0	0	0
$ab, ac$	7	$\frac{7}{1000}$	$\frac{3}{23}$
$\bar{a}, ab, ac$	25	$\frac{25}{1000}$	$\frac{10}{23}$

$n = 1000$

$\llbracket e \rrbracket$	$\#_{0.2}$	$\#_{0.8}$	$\#_{0.5}$
$\perp$	0	0	0
$\diamond$	24	28	23
$\bar{a}$	647	632	614
$ab$	66	246	165
$ac$	231	59	169
$\bar{a}, ab$	0	0	0
$\bar{a}, ac$	0	0	0
$ab, ac$	7	8	4
$\bar{a}, ab, ac$	25	27	25

Table 3: *Experiment 1: bias to ac.* Results from an experiment where  $n = 1000$  samples were generated following the *Model+Noise* procedure with parameters  $\alpha = 0.1, \beta = 0.3, \gamma = 0.2$ . The *empirical* distribution is represented by the random variable  $S$  while the *prior*, as before, is denoted by  $E$ .

where

$$\begin{cases} p & x \\ & y \end{cases}$$

denotes “the value of  $x$  with probability  $p$ , otherwise  $y$ ” — notice that  $y$  might entail  $x$  and *vice-versa*: E.g. some  $ab$  can be generated in the RCE.

- *Other Processes.* Besides the two sample generations procedures above, any other processes and variations can be used. For example, requiring that one of  $x, \bar{x}$  literals is always in a sample or using specific distributions to guide the sampling of literals or events.

Table 4: *Experiments 2 and 3.* Results from experiments, each with  $n = 1000$  samples generated following the *Model+Noise* procedure, with parameters  $\alpha = 0.1, \beta = 0.3, \gamma = 0.8$  (Experiment 2: bias to  $ab$ .) and  $\gamma = 0.5$  (Experiment 3: balanced  $ab$  and  $ac$ .) Empirical distributions are represented by the random variables  $S_{0.8}$  and  $S_{0.5}$  respectively. Data from experience table 3 is also included, and denoted by  $S_{0.2}$ , to provide reference. Columns  $\#_{0.2}, \#_{0.8}$  and  $\#_{0.5}$  contain  $\#\{S_{0.2} \in [e]_{\sim}\}, \#\{S_{0.8} \in [e]_{\sim}\}$  and  $\#\{S_{0.5} \in [e]_{\sim}\}$ , the respective number of events in each class.

**Relating the Empirical and the Prior Distributions.** The data from the simulated observations is used to test the prior distribution. Consider the prior,  $P(E)$ , and the empirical,  $P(S)$ , distributions and the following error function:

$$\text{err}(\theta) := \sum_{e \in \mathcal{E}} (P(E = e) - P(S = e))^2. \quad (19)$$

Since  $E$  depends on  $\theta$ , one can ask how does the error varies with  $\theta$ , what is the *optimal* (i.e. minimum) error value

$$\hat{\theta} := \arg \min_{\theta} \text{err}(\theta) \quad (20)$$



and what does it tell us about the program.

In order to illustrate this analysis, consider the experiment summarized in table 3.

1. Equation (19) becomes

$$\text{err}(\theta) = \frac{20869963}{66125000} + \frac{477}{52900}\theta + \frac{18}{529}\theta^2.$$

2. The minimum of  $\text{err}(\theta)$  is at  $\frac{477}{52900} + 2\frac{18}{529}\theta = 0$ . Since this value is negative and  $\theta \in [0, 1]$ , it must be  $\hat{\theta} = 0$ , and

$$\text{err}(\hat{\theta}) = \frac{20869963}{66125000} \approx 0.31561.$$

The parameters  $\alpha, \beta, \gamma$  of that experiment favour ac over ab. In particular, setting  $\gamma = 0.2$  means that in the simulation process, choices between ab and ac favour ac, 4 to 1. For completeness sake, we also describe one experiment that favours ab over ac (setting  $\gamma = 0.8$ ) and one balanced ( $\gamma = 0.5$ ).

For  $\gamma = 0.8$ , the error function is

$$\begin{aligned} \text{err}(\theta) &= \frac{188207311}{529000000} - \frac{21903}{264500}\theta + \frac{18}{529}\theta^2 \\ &\approx 0.35579 - 0.08281\theta + 0.03403\theta^2 \end{aligned}$$

and, with  $\theta \in [0, 1]$  the minimum is at  $-0.08281 + 0.06805\theta = 0$ , i.e.:

$$\hat{\theta} : \frac{0.08281}{0.06805} \approx 1.21683 > 1.$$

So,  $\hat{\theta} = 1$ ,  $\text{err}(\hat{\theta}) \approx 0.30699$ .

For  $\gamma = 0.5$ , the error function is

$$\begin{aligned} \text{err}(\theta) &= \frac{10217413}{33062500} - \frac{2181}{66125}\theta + \frac{18}{529}\theta^2 \\ &\approx 0.30903 - 0.03298\theta + 0.03402\theta^2 \end{aligned}$$

and, with  $\theta \in [0, 1]$  the minimum is at  $-0.03298 + 0.06804\theta = 0$ , i.e.:

$$\begin{aligned} \hat{\theta} &\approx \frac{0.03298}{0.06804} \approx 0.48471 \approx \frac{1}{2}, \\ \text{err}(\hat{\theta}) &\approx 0.30104 \end{aligned}$$

These experiments show that data can indeed be used to estimate the parameters of the model. However, we observe that the estimated  $\hat{\theta}$  has a tendency to over- or under- estimate the  $\theta$  used to generate the samples. More precisely, in experiment 3 data is generated with  $\gamma = 0.2$  (the surrogate of  $\theta$ ) which is under-estimated with  $\hat{\theta} = 0$  while in experiment 2,  $\gamma = 0.8$  leads the over-estimation  $\hat{\theta} = 1$ . This suggests that we might need to refine the error estimation process. However, experiment 3 data results from  $\gamma = 0.5$  and we've got  $\hat{\theta} \approx 0.48471 \approx 0.5$ , which is more in line with what is to be expected.

## A.2 An Example Involving Bayesian Networks

As it turns out, our framework is suitable to deal with more sophisticated cases, in particular cases involving bayesian networks. In order to illustrate this, in this section we see how the classical example of the Burglary, Earthquake, Alarm (Pearl 1988) works in our setting. This example is a commonly used example in bayesian networks because it illustrates reasoning under uncertainty. The gist of the example is given in fig. 3. It involves a simple network of events and conditional probabilities.

The events are: Burglary (B), Earthquake (E), Alarm (A), Mary calls (M) and John calls (J). The initial events B and E are assumed to be independent events that occur with probabilities  $P(B)$  and  $P(E)$ , respectively. There is an alarm system that can be triggered by either of the initial events B and E. The probability of the alarm going off is a conditional probability given that B and E have occurred. One denotes these probabilities, as per usual, by  $P(A | B)$ , and  $P(A | E)$ . There are two neighbors, Mary and John who have agreed to call if they hear the alarm. The probability that they do actually call is also a conditional probability denoted by  $P(M | A)$  and  $P(J | A)$ , respectively.

We follow the convention of representing the (upper case) random variable X by the (lower case) positive literal x. Considering the probabilities given in fig. 3 we obtain the following specification:

$$\begin{aligned} b &: 0.001, \\ e &: 0.002, \end{aligned}$$

For the table giving the probability  $P(M | A)$  we obtain the program:

$$\begin{aligned} p_{m|a} &: 0.9, \\ p_{m|\bar{a}} &: 0.05, \\ m &\leftarrow a \wedge p_{m|a}, \\ m &\leftarrow \neg a \wedge p_{m|\bar{a}}. \end{aligned}$$

The latter program can be simplified (abusing notation) by writing  $m : 0.9 \leftarrow a$  and  $m : 0.05 \leftarrow \neg a$ .

Similarly, for the probability  $P(J | A)$  we obtain

$$\begin{aligned} j &: 0.7 \leftarrow a, \\ j &: 0.01 \leftarrow \neg a, \end{aligned}$$

Finally, for the probability  $P(A | B \wedge E)$  we obtain

$$\begin{aligned} a &: 0.95 \leftarrow b \wedge e, \quad a : 0.94 \leftarrow b \wedge \bar{e}, \\ a &: 0.29 \leftarrow \bar{b} \wedge e, \quad a : 0.001 \leftarrow \bar{b} \wedge \bar{e}. \end{aligned}$$

One can then proceed as in the previous subsection and analyze this example. The details of such analysis are not given here since they are analogous, albeit admittedly more cumbersome.

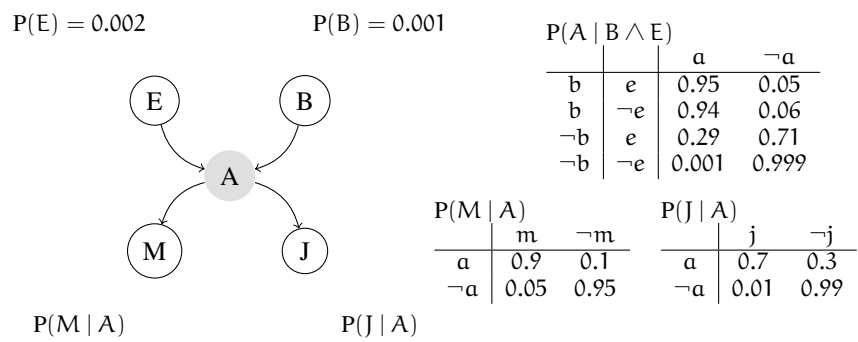


Figure 3: The Earthquake, Burglary, Alarm model