Overall Classification

Originality: 3.33: (fair + 1/3)
Significance: 2.66 (fair - 1/3)
Technical quality 2.66: (poor + 2/3)
Presentation 3.33 (good - 2/3)

1 Review 1

The presentation of the paper is unclear in several areas, making it difficult to fully understand the proposed work. Here are some specific concerns and suggestions for improvement:

1. Grounding and Propositional Case.
2. Negation and Rule Heads: There are issues with the syntax and semantics, particularly concerning negative literals in rule heads. It is unclear whether you are considering "default" negation ("not" as used in systems) or "strong" negation ("-" as used in systems). For example: $a \vee \text{not } a$.
   This should yield two answer sets, {} and $\{a\}$, but the latter is not minimal for the program reduct as defined in the paper. Are you considering $\neg a$ as part of the "empty" answer set? This issue also affects the definition of "stable core" on page 7, where it is claimed that "no stable model contains another," which is not true in the presence of choice rules. Additionally, disjunctive rules are referred to as choice rules, which is misleading in the context of ASP literature. Although choice rules can be converted into a set of normal rules (as claimed on page 3), this does not apply to disjunctive rules used in the paper.
3. Delayed Definitions: The notation $\mathcal{T}$ is used on page 4 but is not defined until page 5. It would be clearer to define all notation when first introduced.
4. Handling Lack of Stable Models: There seems to be a problem with how the approach handles the absence of stable models. Consider the following example:
   $$a : 0.3, b \leftarrow a, \text{not } b.$$
   When a is true, there is no stable model, so the only stable model of the rewritten program is $\{\neg a\}$. Does this model take probability 1 or 0.7? The paper does not clearly explain how such cases, which can become much more intricate, are addressed by the proposed approach.

Originality: 3: (fair)
Significance: 3: (fair)
Technical quality 2: (poor)
Presentation 2: (poor)

franc

- We must be more explicit concerning negations, in particular *default* negation, and about the representation of events and stable models.
- The observation *Handling Lack of Stable Models* suggests to me that we where not clear about the **weights** in annotated atoms; later, we make it worst, when we talk about P_T vs. P_E.

Maybe we should drop the interpretation of annotations in atoms as probabilities and start treating them as weights, from the very start — there's nothing to be lost and we gain clarity and simplicity.

2 Review 2

Positives: The text is well-written, and generally not too hard to read and understand.

However, there are three main reasons that I do not consider this research suitable for presentation at the main ICLP conference.

- Semantics: The usage of stable models is unclear, as they somehow seem to taken as only partly specified (a little bit as if they were well-founded models)?
- Motivation: It is not made sufficiently clear why we need another probabilistic formalism based on answer set programming, or why this particular formalisms should fill that niche.
- No theoretical (or empirical) results that link SASP to other formalisms, or prove objectively important properties to motivate the formalism. In fact, the only results of the paper concern simple properties of the presented formalism.

More specifically: In Example 1, it is claimed that $\{ab\}$ and $\{ac\}$ are stable models of Program (2). This is correct when writing stable models as sets of atoms, where $\{ab\}$ is just a shorthand for $\{ab\neg c\}$. In this case though, the remaining stable model should be written as {}, not as $\{\neg a\}$, and really stands for $\{\neg a, \neg b, \neg c\}$ as sets of literals. It is unclear to me how these stable models arise, and why they are then used later as if they could equally be extended with positive as with negative literals.

In the introduction, the new framework is motivated with the scenarios of supporting probabilistic reasoning tasks and scoring programs with respect to a dataset. However, they do not address how their formalism is better suited to those tasks than other probabilistic logic programming languages that have been developed so far, such as P-Log or LPMLN.

In general, the probability space of events seems strange, and the fact that the probability depends only on the stable core, with no regard to the specificity of the event in question, is rather counter-intuitive. It would be essential to have a discussion of these choices and on why we consider this sigma-algebra of models rather than the sigma-algebra of completely specified models as our probability space.

As a side note, I find the name of the new formalism slightly unfortunate, since it evokes stochastic logic programs, as introduced by Muggleton (Stochastic logic programs. In L. de Raedt (Ed.), Advances in inductive logic programming. Amsterdam: IOS Press 1996). This is a rather special framework with a rather different semantics, and I feel the name should really be reserved for an ASP-analogue of that language. But this may just be a personal thing.

Originality: 4: (good)
Significance: 2: (poor)
Technical quality 2: (poor)
Presentation 4: (good)

franc

- I'm a bit lost with the difference between stable and well-founded models. Anyway, this review led me to realize that we can separate our work in two key steps:
  1. Assign weights (or probabilities) to stable models.
  2. Propagate those to (general) events.

  Although we propose a method to the first step, other methods can be used as well. For example, using P-log or LPMLN.
- Concerning the *Motivation*, I'm not sure how to best address the remark. Using a more elaborate example? Explaining in more detail the scope of our approach? Both?
- It seems that this reviewer got confused about our notation of stable models; And it is rightly so, because we are using a non-standard notation and were not clear about that.
- The point about the space of events and the stable core is also pertinent: "*with no regard to the specificity of the event in question*"; Our interpretation of events needs further framing.

We can use other approaches for the probability of stable models, such as P-log or LPMLN to build upon the probability of events.

3 Review 3

This paper proposes a new approach to probabilistic answer set programming. The motivation is to have a logic program P describe a “system” S: this apparently means a set of observable and latent variables, with each stable model of P a model of state of S. Ideally, the probability of these stable models will be close to the frequency of literals in observable events.

Their approach begins by annotating some atoms in P with probability mass. Each annotated fact A is then transformed into a disjunct with both A and not A annotated. A total stable model S_M then corresponds to a total choice in the sense of the distribution semantics, and the probability of S_M is the probability of the total choice underlying S_M. So far so good.

The next step is to relate an event E (a set of literals) to the set of stable models for P, This is done by defining a stable core of for E as all of the stable models that contain E or are contained by E. The authors then note an induced equivalence on Events where events that have the same stable core are equivalent.

With these observations in mind, the authors specify four functions to determine the probability of an event Evt: μ_T the probability of a total choice T: μ_M, the assignment of probability to each stable model consistent with T: $\mu_C(t)$ is summed probability of the stable models of t in the core of E, which is then normalized for $\mu_C(t)$ for all t.

This is sensible enough and technically correct, as far as I understood it. However, I was left wondering why I should use this approach. The three propositions they prove are all weak, and their running example is pedagogical, but small and abstract. To recommend this paper, I would need better examples that show their the real usefulness of their approach. Do interesting systems have states that actually correspond to stable models? How easy is parameter learning in their approach? This line or research might be fruitful, but I don’t see good evidence to support it in this paper.

Originality: 3: (fair)
Significance: 3: (fair)
Technical quality 4: (good)
Presentation 4: (good)

franc

- “*better examples that show their the real usefulness of their approach*”
- “*How easy is parameter learning in their approach?*”