24th International Joint Conference on Artificial Intelligence IJCAI-15

Proceedings of the 10th International Workshop on Neural-Symbolic Learning and Reasoning NeSy'15

Tarek R. Besold, Luis C. Lamb, Thomas Icard, and Risto Miikkulainen
(eds.)

Buenos Aires, Argentina, 27th of July 2015

Preface

NeSy'15 is the tenth installment of the long-running series of international workshops on Neural-Symbolic Learning and Reasoning (NeSy), which started in 2005 at IJCAI-05 in Edinburgh, Scotland, and also gave rise to two Dagstuhl seminars on the topic, held in 2008 and 2014.

Both, the NeSy workshops and the seminars offer researchers in the areas of artificial intelligence (AI), neural computation, cognitive science, and neighboring disciplines opportunities to get together and share experience and practices concerning topics at the intersection between machine learning and knowledge representation and reasoning through the integration of neural computation and symbolic AI.

Topics of interest for the 2015 edition of NeSy correspondingly include:

- The representation of symbolic knowledge by connectionist systems.
- Neural Learning theory.
- Integration of logic and probabilities, e.g., in neural networks, but also more generally.
- Structured learning and relational learning in neural networks.
- Logical reasoning carried out by neural networks.
- Integrated neural-symbolic learning approaches.
- Extraction of symbolic knowledge from trained neural networks.
- Integrated neural-symbolic reasoning.
- Neural-symbolic cognitive models.
- Biologically-inspired neural-symbolic integration.
- Applications in robotics, simulation, fraud prevention, natural language processing, semantic web, software engineering, fault diagnosis, bioinformatics, visual intelligence, etc.

NeSy'15 offers a diverse mixture of topics and formats, with two keynote lectures by Dan Roth (University of Illinois at Urbana-Champaign) and Gary F. Marcus (New York University), two invited papers from the IJCAI-15 main conference, and four contributed papers to the workshop itself ranging from technical to fairly philosophical considerations concerning different aspects relevant to neural-symbolic integration.

We, as workshop organizers, want to thank the following members of the NeSy'15 program committee for their time and efforts in reviewing the submissions to the workshop and providing valuable feedback to accepted and rejected papers alike:

- Artur D'Avila Garcez, City University London, UK
- Ross Gayler, Melbourne, Australia
- Ramanathan V. Guha, Google Inc., U.S.A.
- Pascal Hitzler, Wright State University, U.S.A.
- Steffen Hoelldobler, Technical University of Dresden, Germany
- Frank Jaekel, University of Osnabrueck, Germany
- Kai-Uwe Kuehnberger, University of Osnabrueck, Germany
- Christopher Potts, Stanford University, U.S.A.
- Ron Sun, Rensselaer Polytechnic Insitute, U.S.A.
- Jakub Szymanik, University of Amsterdam, The Netherlands
- Gerson Zaverucha, Federal University of Rio de Janeiro, Brazil

These workshop proceedings are available online from the workshop webpage under http://www.neural-symbolic.org/NeSy15/.

Osnabrück, 8th of July 2015
Tarek R. Besold, Luis C. Lamb, Thomas Icard, and Risto Miikkulainen.[1]

[1] Tarek R. Besold is a postdoctoral researcher in the AI Group at the Institute of Cognitive Science of the University of Osnabrück, Germany; Luis C. Lamb is Professor of Computer Science at UFRGS, Porto Alegre, Brazil; Thomas Icard is Assistant Professor of Philosophy and Symbolic Systems at Stanford University, USA; Risto Miikkulainen is Professor of Computer Science and Neuroscience at the University of Texas at Austin, USA.

Contents: Contributed Papers

Same same, but different? A research program exploring differences in complexity between logic and neural networks

Tarek R. Besold
Institute of Cognitive Science, University of Osnabrück
Osnabrück, Germany
tarek.besold@uos.de

Abstract

After an overview of the status quo in neural-symbolic integration, a research program targeting foundational differences and relationships on the level of computational complexity between symbolic and sub-symbolic computation and representation is outlined and proposed to the community.

1 Integrating symbolic and sub-symbolic computation and representation

A seamless coupling between learning and reasoning is commonly taken as basis for intelligence in humans and, in close analogy, also for the biologically-inspired (re-)creation of human-level intelligence with computational means (see, e.g., [Valiant, 2013], p. 163). Still, one of the unsolved methodological core issues in human-level AI, cognitive systems modelling, and cognitive and computational neuroscience is the question of the integration between connectionist sub-symbolic (i.e., "neural-level") and logic-based symbolic (i.e., "cognitive-level") approaches to representation, computation, (mostly sub-symbolic) learning, and (mostly symbolic) higher-level reasoning.

AI researchers working on the modelling or (re-)creation of human cognition and intelligence, and cognitive neuroscientists trying to understand the neural basis for human cognition, have for years been interested in the nature of brain-computation in general (see, e.g., [Adolphs, 2015]) and the relation between sub-symbolic/neural and symbolic/cognitive modes of representation and computation in particular (see, e.g., [Dinsmore, 1992]). The brain has a neural structure which operates on the basis of low-level processing of perceptual signals, but cognition also exhibits the capability to efficiently perform abstract reasoning and symbol processing; in fact, processes of the latter type seem to form the conceptual cornerstones for thinking, decision-making, and other (also directly behavior-relevant) mental activities (see, e.g., [Fodor and Pylyshyn, 1988]).

Building on these observations – and taking into account that hybrid systems loosely combining symbolic and sub-symbolic modules into one architecture turned out to be insufficient – agreement on the need for fully integrated neural-cognitive processing has emerged (see, e.g., [Bader and Hitzler, 2005; d'Avila Garcez *et al.*, 2015]). This has several reasons also beyond the analogy to the described functioning principles of the brain:

- In general, network-based approaches possess a higher degree of biological motivation than symbol-based approaches, also outmatching the latter in terms of learning capacities, robust fault-tolerant processing, and generalization to similar input. Also, in AI applications they often enable flexible tools (e.g., for discovering and processing the internal structure of possibly large data sets) and efficient signal-processing models (which are biologically plausible and optimally suited for a wide range of applications).
- Symbolic representations are generally superior in terms of their interpretability, the possibilities of direct control and coding, and the extraction of knowledge when compared to their (in many ways still black box-like) connectionist counterparts.
- From a cognitive modelling point of view, sub-symbolic representations for tasks requiring symbolic high-level reasoning might help solving, among many others, the problem with "too large" logical (epistemic) models (see, e.g., [Gierasimczuk and Szymanik, 2011]) which seem to lead to implausible computations from the reasoning agent's perspective as documented, among others, by [Degremont *et al.*, 2014].

Concerning our current understanding of the relationship and differences between symbolic and sub-symbolic computation and representation, the cognitive-level "symbolic paradigm" is commonly taken to correspond to a Von Neumann architecture (with predominantly discrete and serial computation and localized representations) and the neural-level "sub-symbolic paradigm" mainly is conceptualized as a dynamical systems-type approach (with distributed representations and predominantly parallel and continuous computations).

This divergence notwithstanding, both symbolic/cognitive and sub-symbolic/neural models are considered from a computability perspective equivalent in practice [Siegelmann, 1999], and from a tractability perspective in practice [van Rooij, 2008] as well as likely in theory [van Emde Boas, 1990]. Also, [Leitgeb, 2005] showed that *in principle* there is no substantial difference in representational or problem-solving power between both paradigms (see Sect. 2 for fur-

ther discussion of the cited results).

Still, in general experiences from application studies consistently and reliably show different degrees of suitability and performance of the paradigms in different types of application scenarios, with sub-symbolic/neural approaches offering themselves, e.g., for effective and efficient solutions to tasks involving learning and generalization, while high-level reasoning and concept composition are commonly addressed in symbolic/cognitive frameworks. Unfortunately, general explanations (and solutions) for this foundational dichotomy this far have been elusive when using standard methods of investigation.

In summary, symbolic/cognitive interpretations of artificial neural network (ANN) architectures and accurate and feasible sub-symbolic/neural models of symbolic/cognitive processing seem highly desirable: as important step towards the computational (re-)creation of mental capacities, as possible sources of an additional (bridging) level of explanation of cognitive phenomena of the human brain (assuming that suitably chosen ANN models correspond in a meaningful way to their biological counterparts), and also as important part of future technological developments (also see Sect. 6). But while there is theoretical evidence indicating that both paradigms indeed share deep connections, how to explicitly establish and exploit these correspondences currently remains a mostly unsolved question.

2 Status quo in neural-symbolic integration

Neural-symbolic integration in AI, cognitive modelling, and machine learning

Research on integrated neural-symbolic systems (especially in AI and to a certain extent also in cognitive modelling) has made significant progress over the last two decades (see, e.g., [Bader and Hitzler, 2005; d'Avila Garcez *et al.*, 2015]); partially, but not exclusively, in the wake of the development of deep learning approaches to machine learning (see, e.g., [Schmidhuber, 2015]. Generally, what seem to be several important steps towards the development of integrated neural-symbolic models have been made:

- From the symbolic perspective on the capacities of sub-symbolic computation and representation, the "Propositional Fixation" (i.e., the limitation of neural models on implementing propositional logic at best) has been overcome, among others, in models implementing modal or temporal logics with ANNs (see, e.g., [d'Avila Garcez *et al.*, 2008]).
- From the sub-symbolic perspective, neural computation has been equipped with features previously (almost) exclusively limited to symbolic models by adding top-down governing mechanisms to modular, neural learning architectures, for example, through the use of "Conceptors" [Jaeger, 2014] as computational principle.
- Deep learning approaches to machine learning – by the high number of parameterized transformations performed in the corresponding hierarchically structured models – seem to, at first sight, also conceptually provide what can be interpreted as different levels of abstraction above and beyond mere low-level processing. The resulting networks partially perform tasks classically involving complex symbolic reasoning such as, for instance, the labeling of picture elements or scene description (see, e.g., [Karpathy and Li, 2014; Vinyals *et al.*, 2014]).
- Recently proposed classes of sub-symbolic models such as "Neural Turing Machines" [Graves *et al.*, 2014] or "Memory Networks" [Weston *et al.*, 2015] seem to also architecturally narrow the gap between the (sub-symbolic) dynamical systems characterization and the (symbolic) Von Neumann architecture understanding.

Nonetheless, all these developments (including deep neural networks as layered recurrent ANNs) stay within the possibilities and limitations of the respective classical paradigms without significantly changing the basic formal characteristics of the latter.

Formal analysis of symbolic and sub-symbolic computation and representation

According to our current knowledge, from a formal perspective – especially when focusing on actually physically-realizable and implementable systems (i.e., physical finite state machines) instead of strictly abstract models of computation, together with the resulting physical and conceptual limitations – both symbolic/cognitive and sub-symbolic/neural models seem basically equivalent.

As already mentioned in Sect. 1, notwithstanding partially differing theoretical findings and discussions (as, e.g., given by [Tabor, 2009]), both paradigms are considered computability-equivalent in practice [Siegelmann, 1999]. Also from a tractability perspective, for instance in [van Rooij, 2008], equivalence in practice with respect to classical dimensions of analysis (i.e., interchangeability except for a polynomial overhead) has been established, complementing and supporting the theoretical suggestion of equivalence by [van Emde Boas, 1990] in his Invariance Thesis. Finally, [Leitgeb, 2005] provided an *in principle* existence result, showing that there is no substantial difference in representational or problem-solving power between dynamical systems with distributed representations or symbolic systems with non-monotonic reasoning capabilities.

Still, these results are only partially satisfactory: Although introducing basic connections and mutual dependencies between both paradigms, the respective levels of analysis are quite coarse and the found results are only existential in character. While establishing the *in principle* equivalence described above, [Leitgeb, 2005] does not provide constructive methods for how to actually obtain the corresponding symbolic counterpart to a sub-symbolic model and vice versa.

Concerning the complexity and computability equivalences, while the latter is supported by the results of [Leitgeb, 2005], the former stays mostly untouched: While coming to the same conclusion, i.e., the absence of *substantial* differences between paradigms (i.e., differences at the level of tractability classes), no further clarification or characterization of the precise nature and properties of the polynomial overhead between symbolic and sub-symbolic approaches is provided.

Summary

Although remarkable successes have been achieved within the respective paradigms, the divide between the paradigms persists, interconnecting results still either only address specific and non-generalizable cases or are *in principle* and non-constructive, benchmark scenarios for principled comparisons (e.g., in terms of knowledge representation power or descriptive complexity) between sub-symbolic and symbolic models have still not been established, and questions concerning the precise nature of the relationship and foundational differences between symbolic/cognitive and sub-symbolic/neural approaches to computation and representation still remain unanswered (see, e.g., [Isaac *et al.*, 2014]): in some cases due to a lack of knowledge for deciding the problem, in others due to a lack of tools and methods for properly specifying and addressing the relevant questions.

3 Proposed research program

Focusing on the just described lack of tools and methods, together with the insufficient theoretical knowledge about many aspects of the respective form(s) of computation and representation, in the envisioned research program, the classical findings concerning the relation and integration between the symbolic/cognitive and the sub-symbolic/neural paradigm described in Sect. 1 shall be revisited in light of new developments in the modelling and analysis of connectionist systems in general (and ANNs in particular), and of new formal methods for investigating the properties of general forms of representation and computation on a symbolic level.

To this end, taking into account the apparent empirical differences between the paradigms and (especially when dealing with physically-realizable systems) assuming basic equivalence on the level of computability, emphasis shall be put on identifying and/or developing adequate formal tools and investigating previously unconsidered aspects of existing equivalence results. Focus shall be put on the precise nature of the polynomial overhead as computational-complexity difference between paradigms: Most complexity results for symbolic/cognitive and sub-symbolic/neural computations have been established using exclusively TIME and SPACE as classical resources (see, e.g., [Thomas and Vollmer, 2010; Síma and Orponen, 2003]), and the tractability equivalence between paradigms (see, e.g., [van Rooij, 2008]) mostly leaves out more precise investigations of the remaining polynomial overhead. Against this background, the working hypotheses for the program are that TIME and SPACE are not always adequate and sufficient as resources of analysis for elucidating all relevant properties of the respective paradigms, and that there are significant characteristics and explanations to be found on a more fine-grained level than accessible by classical methods of analysis (settling on the general tractability level).

The main line of research can be summarized in two consecutive questions (corresponding to the stated working hypotheses), one starting out from a more sub-symbolic, the other from a more symbolic perspective:

- **Question 1:** Especially when considering sub-symbolic/neural forms of computation and the associated dynamical systems conception, the adequacy and exhaustiveness of the classical approaches to complexity analysis using only TIME and SPACE as resources for a fully informative characterization must be questioned. Are there more adequate resources which should be taken into account for analysis?
- **Question 2:** Especially when considering the symbolic level, are there more adequate approaches/methods of analysis available than classical complexity theory, allowing to take into account formalism- or calculus-specific characterizations of computations or to perform analyses at a more fine-grained level than tractability?

Finally, in an integrative concluding step taking into account the methods and findings resulting from the previous two, a third question shall be investigated:

- **Question 3:** Can the *in principle* results from [Leitgeb, 2005] be extended to more specific and/or constructive correspondences between individual notions and/or characterizations within the respective paradigms?

Answers to these questions (and the resulting refined tools and methods) promise to contribute to resolving some of the basic theoretical and practical tensions described in Sect. 1 and Sect. 2: Although both paradigms are theoretically undistinguishable (i.e., equivalent up to a polynomial overhead) in their general computational-complexity behavior using classical methods of analysis and characterization results, empirical studies and application cases using state of the art approaches still show clear distinctions in suitability and feasibility of the respective paradigms for different types of tasks and domains without us having an explanation for this behavior. Parts of this divergence might be explained by previously unconsidered and unaccessible complexity-related properties of the respective approaches and their connections to each other.

4 Proposed approach/methodology

In order to address the three leading questions stated in Sect. 3, the program relies on continuous exchange and close interaction with researchers from cognitive science and (computational and cognitive) neuroscience on the one hand, and with logicians and theoretical computer scientists on the other hand. The envisioned level of work is situated between the (purely theoretical) development of methods in complexity theory, network analysis, etc. and the (purely applied) study of properties of computational and representational paradigms by applying existing tools: Previous work from the different fields and lines of research shall be assessed and combined – in doing so, where necessary, adapting or expanding the respective methods and tools – into new means of analysis, which then shall subsequently be applied to suitably selected candidate models representing paradigmatic examples of symbolic or sub-symbolic representations/computations with respect to features relevant for the respective question(s) at hand.

The proposed research program is divided into three stages, corresponding to the three questions from Sect. 3:

Adequate resources for analysis

TIME and SPACE are the standard resources considered in classical complexity analyses of computational frameworks. Correspondingly, most results concerning complexity comparisons between symbolic and sub-symbolic models of computation also focus on these two dimensions (as do, e.g., the aforementioned results by [van Rooij, 2008; van Emde Boas, 1990]).

Still, the reading of TIME and SPACE as mostly relevant resources for complexity analysis is closely connected to a Turing-style conception of computation and a Von Neumann-inspired architecture as machine model, working, e.g., with limited memory. Especially when considering other computational paradigms with different characteristics, as, e.g., the dynamical systems model commonly associated to the sub-symbolic/neural paradigm, the exhaustiveness and adequateness of TIME and SPACE for a full analysis of all relevant computational properties has to be questioned. Instead, it seems likely that additional resources specific to the respective model of computation and architecture have to be taken into account in order to provide a complete characterization.

Thus, in a first stage of the program, popular network types on the sub-symbolic/neural side shall be investigated for relevant dimensions of analysis other than TIME and SPACE. Besides the classical standard and recurrent approaches, models from the the following (non-exhaustive) list could be considered: recurrent spiking neural networks (see, e.g. [Gerstner *et al.*, 2014]), Long Short-Term Memory networks and extensions thereof (see, e.g., [Monner and Reggia, 2012]), or recurrent stochastic neural networks in form of Boltzmann machines [Ackley *et al.*, 1985] and restricted Boltzmann machines [Hinton, 2002].

Taking recurrent networks of spiking neurons as examples, concerning candidates for relevant dimensions also measures such as spike complexity (a bound for the total number of spikes during computation; [Uchizawa *et al.*, 2006]), convergence speed (from some initial network state to the stationary distribution; [Habenschuss *et al.*, 2013]), sample complexity (the number of samples from the stationary distribution needed for a satisfactory computational output; [Vul *et al.*, 2014]), or network size and connectivity seem relevant.

These and similar proposals for the other network models shall be critically assessed (both theoretically and in computational experiments, testing hypotheses and validating the relevance of theoretical results) and, where possible, put into a correspondence relation with each other, allowing to meaningfully generalize between different sub-symbolic/neural models and provide general characterizations of the respective computations.

At the end of this stage, new proposals for adequate resources usable in refined complexity analyses for sub-symbolic/neural computation, together with application examples in terms of proof of concept analyses of popular paradigms, will be available.

Adequate methods of analysis

In parallel to and/or following the search for more adequate resources for complexity analyses of mostly sub-symbolic/neural models of computation, in a second stage of the program emphasis shall be shifted towards the symbolic/cognitive side. While staying closer to the classical conception of complexity in terms of TIME and SPACE, recent developments in different fields of theoretical computer science shall be combined into tools for more model-specific and fine-grained analyses of computational properties.

Parameterized-complexity theory (see, e.g., [Downey and Fellows, 1999]) makes the investigation of problem-specific complexity characteristics possible, while tools such as, e.g., developed in the theory of proof-complexity (see, e.g., [Krajícek, 2005]) allow for for more varied formalism- or calculus-specific characterizations of the respective computations than currently done. Additionally, tools from descriptive complexity theory (see, e.g., [Immerman, 1999]) and work from model-theoretic syntax (see, e.g., [Rabin, 1965]) seem likely to offer chances for shedding light on complexity distinctions below the tractability threshold (i.e., for exploring the precise nature of the polynomial overhead) and to allow for more fine-grained and discriminative comparisons between paradigms and models.

Thus, results from the just mentioned fields/techniques could be examined for their applicability to better characterizing symbolic computation and to potentially establishing conceptual connections to characterizations of sub-symbolic computation from the previous stage:

- Parameterized-complexity theory: Taking into account problem- and application-specific properties of (families of) problems and connecting these to results describing specific properties of sub-symbolic or symbolic computation and representation, trying to explain the different suitability of one or the other paradigm for certain types of tasks.
- Descriptive complexity theory and model-theoretic syntax: Attempting to explore complexity distinctions between different forms of symbolic and between symbolic and sub-symbolic computation also in more fine-grained ways than by mere tractability considerations (e.g., also taking into account the polynomial-time hierarchy and the logarithmic-time hierarchy).
- Proof-complexity theory: Taking into account formalism- and calculus-specific properties of symbolic computations, and trying to map these onto properties of specific sub-symbolic models.

At the end of this stage, proposals for refined methods of analysis for forms of symbolic/cognitive computation and application examples in terms of proof of concept analyses, together with suggestions for correspondences to models of sub-symbolic/neural computation, will be available.

Correspondences between paradigms

In a third and final stage of the program, by combining the results of the previous stages, additional dimensions will be added to previous analyses and established equivalence results, and the precise nature of the polynomial overhead as computational difference between paradigms will be better explained. Also, the outcomes of previous stages shall be integrated where meaningfully possible, ideally providing the foundations for a general set of refined means of analysis for

future comparative investigations of symbolic/cognitive and sub-symbolic/neural computation.

Depending on previous outcomes, some of the following (interrelated) questions probably can be addressed:

- Given the *in principle* equivalence between (symbolic/cognitive) non-monotonic logical systems and (sub-symbolic/neural) dynamical systems, is it possible to establish complexity-based systematic conceptual relationships between particular logical calculi and different types of sub-symbolic networks?
- Can adaptations in network structure and/or (the artificial equivalent of) synaptic dynamics in a neural representation in a systematic way be related to re-representation in a logic-based representation, or (alternatively) is there a systematic correspondence on the level of change of calculus?
- Can changes in network type in a neural representation in a systematic way be related to changes of non-monotonic logic in a symbolic representation?
- Can the correspondences and differences between novel network models approximating classical symbolic capacities (as, e.g., top-down control) or architectures (as, e.g., a Von Neumann machines) and the original symbolic concepts be characterized in a systematic way?

At the end of this stage, partial answers to some of the stated questions, together with proposals for future lines of investigation continuing the work started in the program, will be available. Also, suggestions for new tools and methods for the comparative analysis of symbolic/cognitive and sub-symbolic/neural computation will be available.

5 Expected results/outcomes

More adequate and refined tools and methods for relating and comparing paradigms: New methodological approaches and updated and refined formal tools for better and more adequately analyzing and characterizing the nature and mechanisms of representation and computation in the corresponding paradigm(s) will be developed.

Alternative resources complementing TIME and SPACE for the characterization of properties of (especially sub-symbolic/neural) computation will be provided. Emphasis will be put on making model-specific properties of the respective computing mechanisms accessible.

Also, alternative methods complementing the classical complexity-theoretical approach to the characterization of properties of (especially symbolic/cognitive) computation will be provided. Emphasis will be put on making formalism- or calculus-specific properties of the respective computing mechanisms accessible, and on offering more fine-grained insights than available in the classical framework.

Whilst by itself being useful in more theoretical work, the results shall be maximally informative and usable in the context of neural-cognitive integration in AI, cognitive and computational neuroscience, and (computational) cognitive science.

Principled correspondences between specific notions/characterizations of paradigms: New perspectives on the relation between symbolic/cognitive and sub-symbolic/neural representation and computation will be explored and a better understanding of the respective approach(es) and their interaction (with a strong orientation towards a future integration of conceptual paradigms, of levels of explanation, and of involved scientific disciplines) shall be established. Emphasis will be put on understanding the interaction between model-specific changes in one paradigm and corresponding adaptations of the respective conceptual or formal counterpart within the other paradigm.

6 Potential impact of the proposed program

Integrating symbolic/cognitive and sub-symbolic/neural paradigms of computation and representation not only helps to solve foundational questions within and to strengthen the interface between AI/computer science and cognitive and computational neuroscience, but will also have lasting impact in present and future technological applications and significant possibilities of industrial valorization:

Following the advent of the internet/WWW, ubiquitous sensors and ambient intelligence systems performing high-level and complex reasoning based on low-level data and signals will be key to the future development of advanced intelligent applications and smart environments. Whilst "Big Data" and statistical reasoning can provide for current applications, many real-world scenarios in the near future will require reliable reasoning also based on smaller samples of data, either due to the need for immediate (re)action without the time delay or effort required for obtaining additional usable data, or due to the need of dealing with rare events offering too few similar data entries as to allow for the application of standard statistic-driven approaches. The corresponding systems will, thus, have to make use of complex abstract reasoning mechanisms (which then will have to be used to inform subsequent low-level sensing and processing steps in an action-oriented continuous sensing–processing–reasoning cycle).

This still poses enormous challenges in terms of technological realizability due to the remaining significant divide between symbolic and sub-symbolic paradigms of computation and representation. Here, a better understanding of the relationship between the paradigms and their precise differences in computational and representational power will open up the way to a better integration between both.

References

[Ackley *et al.*, 1985] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski. Learning and Relearning in Boltzmann Machines. *Cognitive Science*, 9(1):147–169, 1985.

[Adolphs, 2015] R. Adolphs. The unsolved problems of neuroscience. *Trends in Cognitive Science*, 19(4):173–175, 2015.

[Bader and Hitzler, 2005] S. Bader and P. Hitzler. Dimensions of Neural-symbolic Integration - A Structured Survey. In *We Will Show Them! Essays in Honour of Dov Gabbay, Volume One*, pages 167–194. College Publications, 2005.

[d'Avila Garcez *et al.*, 2008] A. S. d'Avila Garcez, L. C. Lamb, and D. M. Gabbay. *Neural-Symbolic Cognitive Reasoning*. Cognitive Technologies. Springer, 2008.

[d'Avila Garcez *et al.*, 2015] A. S. d'Avila Garcez, T. R. Besold, L. de Raedt, P. Földiak, P. Hitzler, T. Icard, K.-U. Kühnberger, L. Lamb, R. Miikkulainen, and D. Silver. Neural-Symbolic Learning and Reasoning: Contributions and Challenges. In *AAAI Spring 2015 Symposium on Knowledge Representation and Reasoning: Integrating Symbolic and Neural Approaches*, volume SS-15-03 of *AAAI Technical Reports*. AAAI Press, 2015.

[Degremont *et al.*, 2014] C. Degremont, L. Kurzen, and J. Szymanik. Exploring the tractability border in epistemic tasks. *Synthese*, 191(3):371–408, 2014.

[Dinsmore, 1992] J. Dinsmore, editor. *The Symbolic and Connectionist Paradigms: Closing the Gap*. Cognitive Science Series. Psychology Press, 1992.

[Downey and Fellows, 1999] R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 1999.

[Fodor and Pylyshyn, 1988] J. A. Fodor and Z. W. Pylyshyn. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1–2):3 – 71, 1988.

[Gerstner *et al.*, 2014] W. Gerstner, W. Kistler, R. Naud, and L. Paninski. *Neuronal Dynamics: From Single Neurons to to Networks and Models of Cognition*. Cambridge University Press, 2014.

[Gierasimczuk and Szymanik, 2011] N. Gierasimczuk and J. Szymanik. A Note on a Generalization of the Muddy Children Puzzle. In *Proc. of the 13th Conference on Theoretical Aspects of Rationality and Knowledge*, TARK XIII, pages 257–264. ACM, 2011.

[Graves *et al.*, 2014] A. Graves, G. Wayne, and I. Danihelka. Neural Turing Machines. arXiv. 2014. 1410.5401v1 [cs.NE], 20 Oct 2014.

[Habenschuss *et al.*, 2013] S. Habenschuss, Z. Jonke, and W. Maass. Stochastic computations in cortical microcircuit models. *PLOS Computational Biology*, 9(11), 2013.

[Hinton, 2002] G. E. Hinton. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[Immerman, 1999] N. Immerman, editor. *Descriptive Complexity*. Texts in Computer Science. Springer, 1999.

[Isaac *et al.*, 2014] A. Isaac, J. Szymanik, and R. Verbrugge. Logic and Complexity in Cognitive Science. In *Johan van Benthem on Logic and Information Dynamics*, volume 5 of *Outstanding Contributions to Logic*, pages 787–824. Springer, 2014.

[Jaeger, 2014] H. Jaeger. Controlling recurrent neural networks by conceptors. arXiv. 2014. 1403.3369v1 [cs.CV], 13 Mar 2014.

[Karpathy and Li, 2014] A. Karpathy and F.-F. Li. Deep Visual-Semantic Alignments for Generating Image Descriptions. arXiv. 2014. 1412.2306v1 [cs.CV], 7 Dec 2014.

[Krajícek, 2005] J. Krajícek. Proof Complexity. In *4ECM Stockholm 2004*. European Mathematical Society, 2005.

[Leitgeb, 2005] H. Leitgeb. Interpreted dynamical systems and qualitative laws: From neural networks to evolutionary systems. *Synthese*, 146:189–202, 2005.

[Monner and Reggia, 2012] D. Monner and J. Reggia. A generalized LSTM-like training algorithm for second-order recurrent neural networks. *Neural Networks*, 25:70–83, 2012.

[Rabin, 1965] M. Rabin. A simple method for undecidability proofs and some applications. In Y. Bar-Hillel, editor, *Logic Methodology and Philosophy of Science II*, pages 58–68. North-Holland, 1965.

[Schmidhuber, 2015] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[Siegelmann, 1999] H. Siegelmann. *Neural Networks and Analog Computation: Beyond the Turing Limit*. Birkhäuser, 1999.

[Síma and Orponen, 2003] J. Síma and P. Orponen. General-Purpose Computation with Neural Networks: A Survey of Complexity Theoretic Results. *Neural Computation*, 15:2727–2778, 2003.

[Tabor, 2009] W. Tabor. A dynamical systems perspective on the relationship between symbolic and non-symbolic computation. *Cognitive Neurodynamics*, 3(4):415–427, 2009.

[Thomas and Vollmer, 2010] M. Thomas and H. Vollmer. Complexity of Non-Monotonic Logics. *Bulletin of the EATCS*, 102:53–82, October 2010.

[Uchizawa *et al.*, 2006] K. Uchizawa, R. Douglas, and W. Maass. On the computational power of threshold circuits with sparse activity. *Neural Computation*, 18(12):2994–3008, 2006.

[Valiant, 2013] L. Valiant. *Probably Approximately Correct: Nature's Algorithms for Learning and Prospering in a Complex World*. Basic Books, 2013.

[van Emde Boas, 1990] P. van Emde Boas. Machine Models and Simulations. In *Handbook of Theoretical Computer Science A*. Elsevier, 1990.

[van Rooij, 2008] I. van Rooij. The Tractable Cognition Thesis. *Cognitive Science*, 32:939–984, 2008.

[Vinyals *et al.*, 2014] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and Tell: A Neural Image Caption Generator. arXiv. 2014. 1411.4555v1 [cs.CV], 17 Nov 2014.

[Vul *et al.*, 2014] E. Vul, N. Goodman, , T. Griffiths, and J. Tenenbaum. One and done? Optimal decisions from very few samples. *Cognitive Science*, 38(4):599–637, 2014.

[Weston *et al.*, 2015] J. Weston, S. Chopra, and A. Bordes. Memory Networks. arXiv. 2015. 1410.3916v6 [cs.AI], 7 Feb 2015.

Probabilistic Inference Modulo Theories

Rodrigo de Salvo Braz
SRI International

Ciaran O'Reilly
SRI International

Vibhav Gogate
U. of Texas at Dallas

Rina Dechter
U. of California, Irvine

Abstract

We present SGDPLL(T), an algorithm that solves (among many other problems) probabilistic inference modulo theories, that is, inference problems over probabilistic models defined via a logic theory provided as a parameter (currently, equalities and inequalities on discrete sorts). While many solutions to probabilistic inference over logic representations have been proposed, SGDPLL(T) is the only one that is simultaneously (1) lifted, (2) exact and (3) modulo theories, that is, parameterized by a background logic theory. This offers a foundation for extending it to rich logic languages such as data structures and relational data. By lifted, we mean that our proposed algorithm can leverage first-order representations to solve some inference problems in constant or polynomial time in the domain size (the number of values that variables can take), as opposed to exponential time offered by propositional algorithms.

1 Introduction

Uncertainty representation, inference and learning are important goals in Artificial Intelligence. In the past few decades, neural networks and graphical models have made much progress towards those goals, but even today their main methods can only support very simple types of representations (such as tables and weight matrices) that exclude logical constructs such as relations, functions, arithmetic, lists and trees. Moreover, such representations require models involving discrete variables to be specified at the level of their individual values, making generic algorithms expensive for finite domains and impossible for infinite ones.

For example, consider the following conditional probability distributions, which would need to be either automatically expanded into large tables (a process called *propositionalization*), or manipulated in a manual, ad hoc manner, in order to be processed by mainstream neural networks or graphical model algorithms:

- $P(x > 10 \,|\, y \neq 98 \vee z \leq 15) = 0.1$, for $x, y, z \in \{1, \ldots, 1000\}$
- $P(x \neq Bob \,|\, friends(x, Ann)) = 0.3$

The Statistical Relational Learning [Getoor and Taskar, 2007] literature offered more expressive languages but relied on conversion to conventional representations to perform inference, which can be very inefficient. To counter this, lifted inference [Poole, 2003; de Salvo Braz, 2007] offered solutions for efficiently processing logically specified models, but with languages of limited expressivity (such as function-free ones) and algorithms that are hard to extend. Probabilistic programming [Goodman *et al.*, 2012] has offered inference on full programming languages, but relies on approximate methods on the propositional level.

We present SGDPLL(T), an algorithm that solves (among many other problems) probabilistic inference on models defined over logic representations. Importantly, the algorithm is agnostic with respect to which particular logic theory is used, which is provided to it as a parameter. In this paper, we use the theory consisting of equalities over finite discrete types, and inequalities over bounded integers, as an example. However, SGDPLL(T) offers a foundation for extending it to richer theories involving relations, arithmetic, lists and trees. While many algorithms for probabilistic inference over logic representations have been proposed, SGDPLL(T) is distinguished by being the only existing solution that is simultaneously (1) lifted, (2) exact and (3) modulo theories.

SGDPLL(T) is a generalization of the Davis-Putnam-Logemann-Loveland (DPLL) algorithm for solving the satisfiability problem. SGDPLL(T) generalizes DPLL in three ways: (1) while DPLL only works on propositional logic, SGDPLL(T) takes (as mentioned) a logic theory as a parameter; (2) it solves many more problems than satisfiability on boolean formulas, including summations over real-typed expressions, and (3) it is *symbolic*, accepting input with free variables (which can be seen as constants with unknown values) in terms of which the output is expressed.

Generalization (1) is similar to the generalization of DPLL made by Satisfiability Modulo Theories (SMT) [Barrett *et al.*, 2009; de Moura *et al.*, 2007; Ganzinger *et al.*, 2004], but SMT algorithms require only satisfiability solvers of their theory parameter to be provided, whereas SGDPLL(T) may require solvers for harder tasks (including model counting) that depend on the theory, including *symbolic model counters*, i.e., Figures 1 and 2 illustrate how both DPLL and SGDPLL(T) work and highlight their similarities and differences.

2 Intuition: DPLL, SMT and SGDPLL(T)

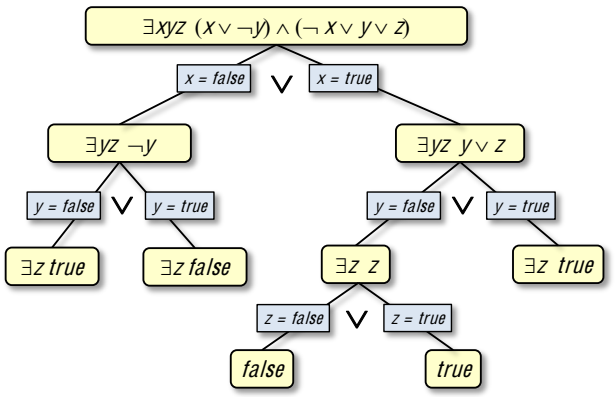

Figure 1: Example of DPLL's search tree for the existence of satisfying assignments. We show the full tree even though the search typically stops when the first satisfying assignment is found.

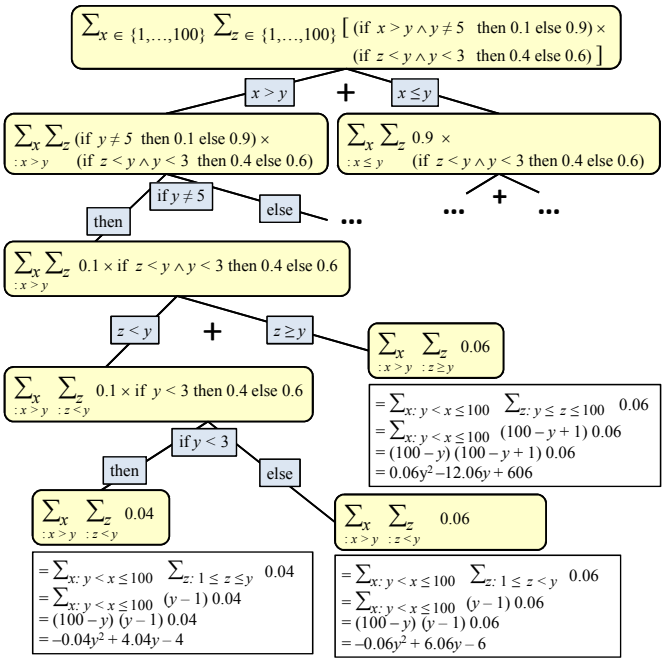

Figure 2: SGDPLL(T) for summation with a background theory of inequalities on bounded integers. It splits the problem according to literals in the background theory, simplifying it until the sum is over a literal-free expression. Some of the splits are on a free variable (y) and create if-then-else expressions which are symbolic conditional solutions. Other splits are on quantified variables (x, z), and split the corresponding quantifier. When the base case with a literal-free expression is obtained, the specific theory solver computes its solution (white boxes). This figure does not show how the conditional subsolutions are summed together; see end of Section 4.1 for examples and details.

The **Davis-Putnam-Logemann-Loveland (DPLL)** algorithm [Davis *et al.*, 1962] solves the **satisfiability** (or **SAT**) problem. SAT consists of determining whether a propositional formula F, expressed in conjunctive normal form (CNF) has a solution or not. A CNF is a conjunction ($\wedge$) of clauses where a clause is a disjunction ($\vee$) of literals, where a literal is a proposition (e.g., x) or its negation (e.g., $\neg x$). A solution to a CNF is an assignment of values from the set $\{0, 1\}$ or $\{\text{TRUE}, \text{FALSE}\}$ to all Boolean variables (or propositions) in F such that at least one literal in each clause in F is assigned to TRUE.

Algorithm 1 shows a simplified and non-optimized version of DPLL which operates on CNF formulas. It works by recursively trying assignments for each proposition, one at a time, simplifying the CNF, and terminating if F is a constant (TRUE or FALSE). Figure 1 shows an example of the execution of DPLL. Although simple, DPLL is the basis for modern SAT solvers which improve it by adding sophisticated techniques and optimizations such as unit propagation, watch literals, and clause learning [Eén and Sörensson, 2003].

Algorithm 1 A version of the DPLL algorithm.

DPLL(F)

F: a formula in CNF.
simplify: simplifies boolean formulas given a condition (e.g., $simplify(x \wedge y | \neg y) = \text{FALSE}$)

```
1  if F is a boolean constant
2      return F
3  else v ← pick a variable in F
4      Sol1 ← DPLL(simplify(F| v))
5      Sol2 ← DPLL(simplify(F|¬v))
6      return Sol1 ∨ Sol2
```

Satisfiability Modulo Theories (SMT) algorithms [Barrett *et al.*, 2009; de Moura *et al.*, 2007; Ganzinger *et al.*, 2004] generalize DPLL and can determine the satisfiability of a Boolean formula expressed in first-order logic, where some function and predicate symbols have additional interpretations. Examples of predicates include equalities, inequalities, and uninterpreted functions, which can then be evaluated using rules of real arithmetic. SMT algorithms condition on the *literals* of a background theory T, looking for a truth assignment to these literals that satisfies the formula. While a SAT solver is free to condition on a proposition, assigning it to either TRUE or FALSE regardless of previous choices (truth values of propositions are independent from each other), an SMT solver needs to also check whether a choice for one literal is *consistent* with the previous choices for others, according to T. This is done by a theory-specific model checker provided to the SMT algorithm as a parameter.

SGDPLL(T) is, like SMT algorithms, **modulo theories** but further generalizes DPLL by being **symbolic** and **quantifier-parametric** (thus "Symbolic Generalized DPLL(T)"). These three features can be observed in the prob-

lem being solved by SGDPLL(T) in Figure 2:

$$\sum_{x,z\in\{1,\ldots,100\}} (\texttt{if}\, x > y \wedge y \neq 5\ \texttt{then}\, 0.1\ \texttt{else}\, 0.9) \times (\texttt{if}\, z < y \wedge y < 3\ \texttt{then}\, 0.4\ \texttt{else}\, 0.6)$$

In this example, the problem being solved requires more than propositional logic theory since equality, inequality and other functions are involved. The problem's quantifier is a summation, as opposed to DPLL and SMT's existential quantification $\exists$. Also, the output will be *symbolic* in y because this variable is not being quantified, as opposed to DPLL and SMT algorithms which implicitly assume all variables to be quantified.

Before formally describing SGDPLL(T), we will briefly comment on its three key generalizations.

Quantifier-parametric Satisfiability can be seen as computing the value of an existentially quantified formula; the existential quantifier can be seen as an indexed form of disjunction, so we say it is **based** on disjunction. SGDPLL(T) generalizes SMT algorithms with respect to the problem being solved by computing expressions quantified by ***any* quantifier $\bigoplus$ based on an associative commutative operation** $\oplus$. Examples of ($\bigoplus$, $\oplus$,) pairs are ($\forall$,$\wedge$), ($\exists$,$\vee$), ($\sum$,$+$), and ($\prod$,$\times$). Therefore SGDPLL(T) can solve not only satisfiability (since disjunction is associative commutative), but also validity (using the $\forall$ quantifier), sums, products, model counting, weighted model counting, maximization, and many more.

Modulo Theories SMT generalizes the propositions in SAT to literals in a given theory T, but the theory connecting these literals remains that of boolean connectives. SGDPLL(T) takes a theory $T = (T_{\mathcal{C}}, T_{\mathcal{L}})$, composed of a **constraint theory** $T_{\mathcal{C}}$ and an **input theory** $T_{\mathcal{L}}$. DPLL propositions are generalized to literals in $T_{\mathcal{C}}$ in SGDPLL(T), whereas the boolean connectives are generalized to functions in $T_{\mathcal{L}}$. In the example above, $T_{\mathcal{C}}$ is the theory of inequalities on bounded integers, whereas $T_{\mathcal{L}}$ is the theory of $+$, $\times$, boolean connectives and `if then else`. $T_{\mathcal{L}}$ is used simply for the simplifications after conditioning, which takes time at most linear in the input expression size.

Symbolic Both SAT and SMT can be seen as computing the value of an existentially quantified formula in which *all* variables are quantified, and which is always equivalent to either TRUE or FALSE. SGDPLL(T) further generalizes SAT and SMT by accepting quantifications over *any subset* of the variables in its input expression (including the empty set). The non-quantified variables are **free variables**, and the result of the quantification will typically depend on them. Therefore, SGDPLL(T)'s output is a **symbolic** expression in terms of free variables. Section 3 shows an example of a symbolic solution.

3 T-Problems and T-Solutions

SGDPLL(T) receives a **T-problem** (or, for short, simply a **problem**) of the form

$$\bigoplus_{x_1:C_1} \cdots \bigoplus_{x_m:C_m} E,$$

where, for each $i = 1, \ldots, m$, x_i is an **index variable** quantified by $\bigoplus$ and subject to constraint C_i in $T_{\mathcal{C}}$, and E an expression in $T_{\mathcal{L}}$. C_i is assumed to be equivalent to a conjunction of literals in $T_{\mathcal{C}}$. There may be **free variables**, that is, variables that are not quantified, present in both the constraints and E. An example of a problem is

$$\sum_{y} \sum_{x:3\leq x\wedge x\leq y} \texttt{if}\, x > 4\ \texttt{then}\, y\ \texttt{else}\, 10 + z,$$

for x, y bounded integer variables in, say, $\{1, \ldots, 20\}$.

A **T-solution** (or, for short, simply a **solution**) to a problem is a quantifier-free expression in $T_{\mathcal{L}}$ equivalent to the problem. It can be an **unconditional solution**, containing no literals in $T_{\mathcal{C}}$, or a **conditional solution** of the form $\texttt{if}\, L\ \texttt{then}\, S_1\ \texttt{else}\, S_2$, where L is a literal in $T_{\mathcal{C}}$ and S_1, S_2 are two solutions (either conditional or unconditional). Note that, in order for the solution to be equivalent to the problem, only variables that were free (not quantified) can appear in the literal L. In other words, a solution can be seen as a decision tree on literals, with literal-free expressions in the leaves, such that each leaf is equivalent to the original problem, provided that the literals on the path to it are true. For example, the problem

$$\sum_{x:1\leq x\wedge x\leq 10} \texttt{if}\, y > 2 \wedge w > y\ \texttt{then}\, y\ \texttt{else}\, 4$$

has an equivalent conditional solution

$$\texttt{if}\, y > 2\ \texttt{then}\, \texttt{if}\, w > y\ \texttt{then}\, 10y\ \texttt{else}\, 40\ \texttt{else}\, 40.$$

4 SGDPLL(T)

In this section we provide the details of SGDPLL(T), described in Algorithm 2 and exemplified in Figure 2. We first give an informal explanation guided by examples, and then provide a formal description of the algorithm.

4.1 Informal Description of SGDPLL(T)

Base Case Problems

A problem is in **base case 0** if and only if $m = 1$, E contains no literals in $T_{\mathcal{C}}$ and C is in a **base form** specific to the theory T, which its solver must be able to recognize.

In our running example, we a slight variant of **difference arithmetic** [de Moura *et al.*, 2007], with atoms of the form $x < y$ or $x \leq y + c$, where c is an integer constant. Strict inequalities $x < y + c$ can be represented as $x \leq y + c - 1$ and the negation of $x \leq y + c$ is $y \leq x - c - 1$. From now on, we shorten $a \leq x \wedge x \leq b$ to $a \leq x \leq b$.

The base case for difference arithmetic is $\sum_{x:l\leq x\leq u} E$, where E is a polynomial and x's lower and upper bounds l and u are either variables, or differences between a variable and an integer constant. One example is $\sum_{x:y+1\leq x\leq z-3} y^2 +$

$4x^3$. When $l \leq u$, Faulhaber's formula [Knuth, 1993] allows us to compute a new polynomial E' (in the variables other than x) equivalent to the problem. Moreover, this can be done (a little surprisingly) in time only dependent in the *degree* of the polynomial E, *not* on the domain size of x or the distance $u - l$. [1] If $l \leq u$ is false, there are no values of x satisfying the constraint, and the problem results in 0. Therefore, the solution is the conditional $\texttt{if}\, l \leq u\ \texttt{then}\, E'\ \texttt{else}\, 0$.

A **base case 1** problem is such that $m > 1$ and $\bigoplus_{x:C_m} E$ satisfies base case 0, yielding solution S. In this case, we reduce the problem to the simpler

$$\bigoplus_{x_1:C_1} \cdots \bigoplus_{x_{m-1}:C_{m-1}} S.$$

Non-Base case Problems

For non-base cases, SGDPLL(T) mirrors DPLL, by selecting a **splitter literal** to split the problem on, generating two simpler problems. This eventually leads to base case problems.

The splitter literal L can come from either the expression E, to bring it closer to being literal-free, or from C_m, to bring it closer to satisfying the base form prerequisites. We will see examples shortly.

Once the splitter literal L is chosen, it splits the problem in two possible ways: if L does *not* contain any of the indices x_i, it causes an **if-splitting** in which L is the condition of an `if then else` expression and the two simpler subproblems are its *then* and *else* clauses; if L contains at least one index, it causes a **quantifier-splitting** based on the latest of the indices it contains.

For an example of an if-splitting on a literal coming from C_m, consider the problem

$$\sum_z \sum_{x:y\leq x \wedge 3 \leq x \wedge x \leq 10} y^2.$$

This is not a base case because the constraint includes two lower bounds for x (y and 3), which are not redundant because we do not know which one is the smallest. We can however reduce the problem to base case ones by splitting the problem according to $y < 3$:

$$\texttt{if}\, y < 3\ \texttt{then} \sum_z \sum_{x:3\leq x\leq 10} y^2\ \texttt{else} \sum_z \sum_{x:y\leq x\leq 10} y^2.$$

For another example of an if-splitting, but on a literal coming from E this time, consider

$$\sum_z \sum_{x:3\leq x\leq 10} \texttt{if}\, y > 4\ \texttt{then}\, y\ \texttt{else}\, 10.$$

This is not a base case because E is not literal-free. However, splitting on $y > 4$ reduces to

$$\texttt{if}\, y > 4\ \texttt{then} \sum_z \sum_{x:3\leq x\leq 10} y\ \texttt{else} \sum_z \sum_{x:3\leq x\leq 10} 10,$$

containing two base cases.

[1] This takes a number of steps, which we omit for lack of space, but it is not hard to do.

For an example of quantifier-splitting on a literal coming from E, consider this problem in which the splitter literal contains at least one index (here it contains two, x and z):

$$\sum_{x:3\leq x\leq 10} \sum_z \texttt{if}\, x > 4\ \texttt{then}\, y\ \texttt{else}\, 10 + z.$$

In this case, we cannot simply move the literal outside the scope of the sum in its latest index x. Instead, we add the literal and its negation to the constraint on x, in two new subproblems:

$$\begin{aligned} &= \Big(\sum_{x:x>4\wedge 3\leq x\leq 10} \sum_z y\Big) \quad + \Big(\sum_{x:x\leq 4\wedge 3\leq x\leq 10} \sum_z 10 + z\Big) \\ &= \Big(\sum_{x:5\leq x<10} \sum_z y\Big) \qquad + \Big(\sum_{x:3\leq x\leq 4} \sum_z 10 + z\Big). \end{aligned}$$

In this example, the two sub-solutions are unconditional polynomials, and their sum results in another unconditional polynomial, which is a valid solution. However, if at least one of the sub-solutions is conditional, their direct sum is not a valid solution. In this case, we need to combine them with a distributive transformation of $\oplus$ over `if then else`:

$$\begin{aligned} &(\texttt{if}\, x < 4\ \texttt{then}\, y^2\ \texttt{else}\, z) + (\texttt{if}\, y > z\ \texttt{then}\, 3\ \texttt{else}\, x) \\ &\equiv \texttt{if}\, x < 4\ \texttt{then}\, y^2 + (\texttt{if}\, y > z\ \texttt{then}\, 3\ \texttt{else}\, x) \\ &\qquad\quad \texttt{else}\, z\ + (\texttt{if}\, y > z\ \texttt{then}\, 3\ \texttt{else}\, x) \\ &\equiv \texttt{if}\, x < 4\ \texttt{then}\ \texttt{if}\, y > z\ \texttt{then}\, y^2 + 3\ \texttt{else}\, y^2 + x \\ &\qquad\quad \texttt{else}\ \texttt{if}\, y > z\ \texttt{then}\, z\ + 3\ \texttt{else}\, z\ + x. \end{aligned}$$

4.2 Formal Description of SGDPLL(T)

We now present a formal version of the algorithm. We start by specifying the basic tasks the given T-solver is required to solve, and then show can we can use it to solve any T-problems.

Requirements on T solver

To be a valid input for SGDPLL(T), a T-solver S_T for theory $T = (T_{\mathcal{L}}, T_{\mathcal{C}})$ must solve two tasks:

- Given a problem $\bigoplus_{x:C} E$, S_T must be able to recognize whether C is in base form and, if so, provide a solution $base_T(\bigoplus_{x:C} E)$ for the problem.
- Given a conjunction C *not* in base form, S_T must provide a tuple $split_T(C) = (L, C_L, C_{\neg L})$ such that $L \in T_{\mathcal{C}}$, and conjunctions C_L and $C_{\neg L}$ are smaller than C and satisfy $L \Rightarrow (C \Leftrightarrow C_L) \wedge \neg L \Rightarrow (C \Leftrightarrow C_{\neg L})$.

The algorithm is presented in Figure 2. Note that it does *not* depend on difference arithmetic theory, but can use a **solver for any theory satisfying the requirements above**. Note also that conditional solution may contain **redundant literals**, and that **optimizations** such as unit propagation and watch literals can be employed. These issues can be addressed but we omit the details for lack of space.

If the T-solver implements the operations above in polynomial time in the number of variables and constant time in the domain size (the size of their types), then SGDPLL(T), like DPLL, will have **time complexity** exponential in the number of literals and, therefore, in the number of variables, and

be **independent of the domain size**. This is the case for the solver for difference arithmetic and will be typically the case for many other solvers.

5 Probabilistic Inference Modulo Theories

Let $P(X_1 = x_1, \ldots, X_n = x_n)$ be the joint probability distribution on random variables $\{X_1, \ldots, X_n\}$. For any tuple of indices t, we define X_t to be the tuple of variables indexed by the indices in t, and abbreviate the assignments $(X = x)$ and $(X_t = x_t)$ by simply x and x_t, respectively. Let $\bar{t}$ be the tuple of indices in $\{1, \ldots, n\}$ but not in t.

The **marginal probability distribution** of a subset of variables X_q is one of the most basic tasks in probabilistic inference, defined as $P(x_q) = \sum_{x_{\bar{q}}} P(x)$ which is a summation on a subset of variables occurring in an input expression, and therefore solvable by SGDPLL(T).

If $P(X = x)$ is expressed in the language of input and constraint theories appropriate for SGDPLL(T) (such as the one shown in Figure 2), then it can be solved by SGDPLL(T), *without* first converting its representation to a much larger one based on tables. The output will be a summation-free expression in the assignment variables x_q representing the marginal probability distribution of X_q.

Belief updating consists of computing the **posterior probability** of X_q given evidence on X_e, which is defined as $P(x_q|x_e) = P(x_q, x_e)/P(x_e)$ which can be computed with two applications of SGDPLL(T), one for the marginal $P(x_q, x_e)$ and another for $P(x_e)$.

Applying SGDPLL(T) in the manner above does not take **advantage of factorized representations** of joint probability distributions, a crucial aspect of efficient probabilistic inference. However, it can be used as a basis for an algorithm, Symbolic Generalized Variable Elimination modulo theories **(SGVE(T))**, analogous to Variable Elimination (VE) [Zhang and Poole, 1994] for graphical models, that exploits factorization. Suppose $P(x)$ is represented as a product of real-valued functions (called **factors**) f_i: $P(x) = f_1(x_{t1}) \times \cdots \times f_m(x_{t1})$ and we want to compute a summation over it: $\sum_{x_{\bar{q}}} f_1(x_{t1}) \times \cdots \times f_m(x_{tm})$ where q and t_i are tuples. We now choose a variable x_i for $i \notin q$ to eliminate first. Let g be the product of all factors in which x_i does not appear, h be the product of all factors in which x_i does appear, and b be the tuple of indices of variables other than x_i appearing in h. Then we rewrite the above as

$$\sum_{x_{\bar{q},\bar{i}}} g(x_{\bar{i}}) \sum_{x_i} h(x_i, x_b) = \sum_{x_{\bar{q},\bar{i}}} g(x_{\bar{i}}) h'(x_b)$$

where h' is a summation-free factor computed by SGDPLL(T) and equivalent to the innermost summation. We now have a problem of the same type as originally, but with one less variable, and can proceed until all variables in $x_{\bar{q}}$ are eliminated. SGDPLL(T) being symbolic allows us to compute h' without iterating over all values to x_i.

SGDPLL(T) contributes to the symbolic connectionist literature by providing a way to incorporate logic representations to systems dealing with uncertainty. If the variables and data used by SGDPLL(T) in a given system are of a subsymbolic, distributed nature, then this system will exhibit ad-

Algorithm 2 Symbolic Generalized DPLL (SGDPLL(T)), omitting pruning, heuristics and optimizations.

SGDPLL(T)($\bigoplus_{x_1:C_1} \cdots \bigoplus_{x_m:C_m} E$)

Returns a T-solution for the given T-problem.

if *split*($\bigoplus_{x_m:C_m} E$) indicates "base case"
  $S \leftarrow base_T(\bigoplus_{x_m:C_m} E)$
  if $m = 1$ // decide if base case 0 or 1
    return S
  else
    $P \leftarrow \bigoplus_{x_1:C_1} \cdots \bigoplus_{x_{m-1}:C_{m-1}} S$
else
  // *split* returned $(L, \bigoplus_{x_m:C'_m} E', \bigoplus_{x_m:C''_m} E'')$
  if L does not contain any indices
    $splittingType \leftarrow$ "if"
    $Sub_1 \leftarrow \bigoplus_{x_1:C_1} \cdots \bigoplus_{x_m:C'_m} E'$
    $Sub_2 \leftarrow \bigoplus_{x_1:C_1} \cdots \bigoplus_{x_m:C''_m} E''$
  else // L contains a latest index x_j:
    $splittingType \leftarrow$ "quantifier"
    $Sub_1 \leftarrow \bigoplus_{x_1:C_1} \cdots \bigoplus_{x_j:C_j \wedge L} \cdots \bigoplus_{x_m:C'_m} E'$
    $Sub_2 \leftarrow \bigoplus_{x_1:C_1} \cdots \bigoplus_{x_j:C_j \wedge \neg L} \cdots \bigoplus_{x_m:C''_m} E''$
  $S_1 \leftarrow$ SGDPLL(T)(Sub_1)
  $S_2 \leftarrow$ SGDPLL(T)(Sub_2)
  if $splittingType$ == "if"
    return the expression `if` L `then` S_1 `else` S_2
  else return $combine(S_1, S_2)$

SPLIT($\bigoplus_{x:C} E$)

if E contains a literal L
  $E' \leftarrow E$ with L replaced by TRUE
  $E'' \leftarrow E$ with L replaced by FALSE
  return $(L, \bigoplus_C E', \bigoplus_C E'')$
elseif C is *not* recognized as base form by the T-solver
  $(L, C', C'') \leftarrow split_T(C)$
  return $(L, \bigoplus_{C'} E, \bigoplus_{C''} E)$
else return "base case"

COMBINE(S_1, S_2)

if S_1 is of the form `if` C_1 `then` S_{11} `else` S_{12}
  return the following if-then-else expression:
  `if` C_1
    `then` $combine(S_{11}, S_2)$
    `else` $combine(S_{12}, S_2)$
elseif S_2 is of the form `if` C_2 `then` S_{21} `else` S_{22}
  return the following if-then-else expression:
  `if` C_2
    `then` $combine(S_1, S_{21})$
    `else` $combine(S_1, S_{22})$
else return $S_1 \oplus S_2$

vantages of connectionism, observed on logic representations instead of simpler propositional representations.

6 Evaluation

We did a very preliminary comparison of SGDPLL(T)-based SGVE(T), using an implementation of an equality theory ($=, \neq$ literals only) symbolic model counter, to the state-of-the-art probabilistic inference solver variable elimination and conditioning (VEC) [Gogate and Dechter, 2011], on randomly generated probabilistic graphical models based on equalities formulas, on a Intel Core i7 notebook. We ran both SGVE(T) and VEC on a random graphical model with 10 random variables, 5 factors, with formulas of depth and breadth (number of arguments per sub-formula) 2 for random connectives $\vee$ and $\wedge$. SGVE(T) took 1.5 seconds to compute marginals for all variables (unsurprisingly, irrespective of domain size). We grounded this model for domain size 16 to provide the table-based input required by VEC, which then took 30 seconds to compute all marginals. The largest grounded table given to VEC as input had 6 random variables and therefore around 16 million entries.

7 Related work

SGDPLL(T) is a lifted inference algorithm [Poole, 2003; de Salvo Braz, 2007; Gogate and Domingos, 2011], but the proposed lifted algorithms so far have concerned themselves only with relational formulas with equality. We have not yet developed a relational model counter but presented one or inequalities, the first in the lifted inference literature. SGDPLL(T) generalizes several algorithms that operate on mixed networks [Mateescu and Dechter, 2008] – a framework that combines Bayesian networks with constraint networks, but with a much richer representation.

8 Conclusion and Future Work

We have presented SGDPLL(T) and its derivation SGVE(T), the first algorithms formally able to solve a variety of model counting problems (including probabilistic inference) modulo theories, that is, capable of being extended with theories for richer representations than propositional logic, in a lifted and exact manner. Future work includes additional theories of interest mainly among them those for uninterpreted relations (particularly multi-arity functions) and arithmetic; modern SAT solver optimization techniques such as watched literals and unit propagation anytime approximation schemes that offer guaranteed bounds on approximations that converge to the exact solution.

Acknowledgments

We gratefully acknowledge the support of the Defense Advanced Research Projects Agency (DARPA) Probabilistic Programming for Advanced Machine Learning Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-14-C-0005. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of DARPA, AFRL, or the US government.

References

[Barrett *et al.*, 2009] C. W. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli. Satisfiability Modulo Theories. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 825–885. IOS Press, 2009.

[Davis *et al.*, 1962] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Communications of the ACM*, 5:394–397, 1962.

[de Moura *et al.*, 2007] Leonardo de Moura, Bruno Dutertre, and Natarajan Shankar. A tutorial on satisfiability modulo theories. In *Computer Aided Verification, 19th International Conference, CAV 2007, Berlin, Germany, July 3-7, 2007, Proceedings*, volume 4590 of *Lecture Notes in Computer Science*, pages 20–36. Springer, 2007.

[de Salvo Braz, 2007] R. de Salvo Braz. *Lifted First-Order Probabilistic Inference*. PhD thesis, University of Illinois, Urbana-Champaign, IL, 2007.

[Eén and Sörensson, 2003] N. Eén and N. Sörensson. An Extensible SAT-solver. In *SAT Competition 2003*, volume 2919 of *Lecture Notes in Computer Science*, pages 502–518. Springer, 2003.

[Ganzinger *et al.*, 2004] Harald Ganzinger, George Hagen, Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. *DPLL(T): Fast Decision Procedures*. 2004.

[Getoor and Taskar, 2007] L. Getoor and B. Taskar, editors. *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[Gogate and Dechter, 2011] V. Gogate and R. Dechter. SampleSearch: Importance sampling in presence of determinism. *Artificial Intelligence*, 175(2):694–729, 2011.

[Gogate and Domingos, 2011] V. Gogate and P. Domingos. Probabilistic Theorem Proving. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, pages 256–265. AUAI Press, 2011.

[Goodman *et al.*, 2012] Noah D. Goodman, Vikash K. Mansinghka, Daniel M. Roy, Keith Bonawitz, and Daniel Tarlow. Church: a language for generative models. *CoRR*, abs/1206.3255, 2012.

[Knuth, 1993] Donald E. Knuth. Johann Faulhaber and Sums of Powers. *Mathematics of Computation*, 61(203):277–294, 1993.

[Mateescu and Dechter, 2008] R. Mateescu and R. Dechter. Mixed deterministic and probabilistic networks. *Annals of Mathematics and Artificial Intelligence*, 54(1-3):3–51, 2008.

[Poole, 2003] D. Poole. First-Order Probabilistic Inference. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 985–991, Acapulco, Mexico, 2003. Morgan Kaufmann.

[Zhang and Poole, 1994] N. Zhang and D. Poole. A simple approach to Bayesian network computations. In *Proceedings of the Tenth Biennial Canadian Artificial Intelligence Conference*, 1994.

A Connectionist Network for Skeptical Abduction

Emmanuelle-Anna Dietz and Steffen Hölldobler and Luis Palacios*
International Center for Computational Logic, TU Dresden, 01062 Dresden, Germany
dietz@iccl.tu-dresden.de sh@iccl.tu-dresden.de palacios.medinacelli@gmail.com

Abstract

We present a new connectionist network to compute skeptical abduction. Combined with the CORE method to compute least fixed points of semantic operators for logic programs, the network is a prerequisite to solve human reasoning tasks like the suppression task in a connectionist setting.

1 Introduction

Various human reasoning tasks like the suppression and the selection tasks can be adequately modeled in a computational logic approach based on the weak completion semantics [11; 6; 4; 5]. Therein, knowledge is encoded as logic program and interpreted under the Łukasiwiecz (Ł) logic [18]. As shown in [11] the weak completion of each program admits a least Ł-model and reasoning is performed with respect to these least Ł-models. The least Ł-models can be computed as least fixed points of a particular semantic operator which was introduced in [22]. In addition, some tasks require abduction and, in particular, it has been shown in [12; 6] that skeptical abduction is needed as otherwise, if credulous abduction is applied, the approach is no longer adequate.

As shown in [10], the above mentioned semantic operator can be computed by a three-layer feed-forward network within the CORE method [8; 2], where the input as well as the output layer represent interpretations. By connecting the units in the output layer to the units in the input layer, the corresponding recurrent network converges to a stable state encoding the least fixed point of the semantic operator.

In order to develop a fully connectionist network for the suppression and the selection tasks we need to add abduction to the recurrent networks mentioned in the previous paragraph. Unfortunately, to the best of our knowledge, the only connectionist models for abduction compute credulous conclusions [3]. Hence, there is the need to develop a connectionist model for skeptical abduction and we will do this in this paper for two-valued logic.

2 Preliminaries

We introduce the general notation and terminology that will be used throughout the paper, based on [17; 13] and [14].

A *(propositional logic) program* is a finite set of (program) clauses of the form $A \leftarrow L_1 \wedge \ldots \wedge L_n,\ n \geq 0$, where A is an atom and L_i, $1 \leq i \leq n$, are literals. A is called *head* and $L_1 \wedge \ldots \wedge L_n$ is called *body*. We also refer to the body as the set $\{L_1, \ldots, L_n\}$. If the body is empty we write A. $\mathsf{atoms}(\mathcal{P})$ denotes the set of atoms occurring in $\mathcal{P}$. $\mathsf{def}(\mathcal{P}) = \{A \mid A \leftarrow \mathit{body} \in \mathcal{P}\}$ is the set of *defined* atoms and $\mathsf{undef}(\mathcal{P}) = \mathsf{atoms}(\mathcal{P}) \setminus \mathsf{def}(\mathcal{P})$ is the set of *undefined* atoms in $\mathcal{P}$.

An *interpretation* I is a mapping from the set of ground atoms to $\{\top, \bot\}$. $\mathcal{M}$ is a *model* of $\mathcal{P}$ if it is an interpretation that maps each clause occurring in $\mathcal{P}$ to $\top$. $\mathcal{M}$ is a *minimal model* of $\mathcal{P}$ iff there is no other model $\mathcal{M}'$ s.t. $\mathcal{M}' \subset \mathcal{M}$. If $\mathcal{P}$ has only one minimal model, then this is its *least model*.

The knowledge represented by a logic program $\mathcal{P}$ can be captured by the $T_\mathcal{P}$ operator [1]. Given an interpretation I and a program $\mathcal{P}$, $T_\mathcal{P}$ is defined as $T_\mathcal{P}(I) = \{A \mid A \leftarrow \mathit{body} \in \mathcal{P}$ and $I(\mathit{body}) = \top\}$. For *acceptable* programs [7], $T_\mathcal{P}$ admits a least fixed point denoted by $T_\mathcal{P}^{\mathsf{lfp}}$. Moreover, this fixed point is the least model of $\mathcal{P}$ [7]. We write $\mathcal{P} \models^{\mathsf{lfp}} F$ iff formula F holds in the least fixed point of $T_\mathcal{P}$.

We consider the *abductive framework* $\mathsf{AF} = \langle \mathcal{P}, \mathcal{A}_\mathcal{P}, \mathcal{IC}, \models^{\mathsf{lfp}} \rangle$, where $\mathcal{P}$ is an acceptable program,[1] $\mathcal{A}_\mathcal{P} = \{A \leftarrow \top \mid A \in \mathsf{undef}(\mathcal{P})\}$ is the set of *abducibles* and $\mathcal{IC}$ is a finite set of *integrity constraints*, i.e., expressions of the form $\bot \leftarrow \mathit{body}$. Let $\mathcal{O}$ be a finite set of literals called *observations*. $\mathcal{O}$ is *explained by* $\mathcal{E} \subseteq \mathcal{A}_\mathcal{P}$ iff $\mathcal{P} \not\models^{\mathsf{lfp}} \mathcal{O}$, $\mathcal{P} \cup \mathcal{E} \models^{\mathsf{lfp}} \mathcal{O}$ and $\mathcal{P} \cup \mathcal{E}$ *satisfies* $\mathcal{IC}$. We assume that explanations are minimal, i.e., there is no other explanation $\mathcal{E}' \subset \mathcal{E}$ for $\mathcal{O}$.

F follows skeptically from $\mathcal{P}$, $\mathcal{IC}$ and $\mathcal{O}$ iff $\mathcal{O}$ can be explained given $\mathcal{P}$ and $\mathcal{IC}$, and for all explanations $\mathcal{E}$ for $\mathcal{O}$ we find $\mathcal{P} \cup \mathcal{E} \models^{\mathsf{lfp}} F$. *$F$ follows credulously from $\mathcal{P}$, $\mathcal{IC}$ and $\mathcal{O}$* iff there exists an explanation $\mathcal{E}$ for $\mathcal{O}$ and $\mathcal{P} \cup \mathcal{E} \models^{\mathsf{lfp}} F$. We focus on skeptical abduction, we are not interested in computing skeptical abductive explanations (i.e. explanations which are present in every model of $\mathcal{P} \cup \mathcal{O}$) but instead we want to know the logical consequences of all explanations for $\mathcal{O}$. This is a keypoint and the main contribution of this paper. The reason why we need to restrict ourselves to minimal explanations can be clarified by the following example: Given

*The authors are listed alphabetically. Luis Palacios was supported by the European Master's Program in Computational Logic

[1] This restricion can be lifted allowing $\mathcal{P}$ to be any logic program using the ideas in [16; 20], where AF is transformed into an equivalent AF* wrt *generalized stable model semantics*. The resulting AF* is definite and can be handled by our approach.

Data: $\langle \mathcal{P}, \mathcal{A}_\mathcal{P}, \mathcal{IC}, \models^{\mathsf{lfp}} \rangle$ and $\mathcal{O}$
Result: Set of minimal explanations $\mathcal{M}_\mathcal{E}$ for $\mathcal{O}$
$\mathcal{M}_\mathcal{E} = \emptyset$
for $i = 1 \dots n$ **do**
  1) **if** *forall* $\mathcal{E} \in \mathcal{M}_\mathcal{E}$ $\mathcal{E} \not\subset \mathcal{C}_i$ *and* $\mathcal{C}_i \in Ord(\mathcal{A}_\mathcal{P})$ **then**
    2) **if** $\mathcal{P} \cup \mathcal{C}_i \models^{\mathsf{lfp}} \mathcal{O}$ *and* $\mathcal{P} \cup \mathcal{C}_i$ *satisfies* $\mathcal{IC}$ **then**
      add $\mathcal{C}_i$ to $\mathcal{M}_\mathcal{E}$
    else
      discard $\mathcal{C}_i$
    end
  else
    discard $\mathcal{C}_i$
  end
end

Algorithm 1: The computation of all minimal explanations.

Data: $\langle \mathcal{P}, \mathcal{A}_\mathcal{P}, \mathcal{IC}, \models^{\mathsf{lfp}} \rangle$, $\mathcal{O}$ and $\mathcal{M}_\mathcal{E}$[2]
Result: Sets $\mathcal{S}^+, \mathcal{S}^-$
$j = 0$
$\mathcal{S}^+ = \mathcal{S}^- = \emptyset$
for *each* $\mathcal{E} \in \mathcal{M}_\mathcal{E}$ **do**
  increment j by 1
  $\mathcal{F}_j^+ = \{A \mid \mathcal{P} \cup \mathcal{E} \models^{\mathsf{lfp}} A\}$
  $\mathcal{F}_j^- = \{\neg A \mid \mathcal{P} \cup \mathcal{E} \models^{\mathsf{lfp}} \neg A\}$
end
if $j \geq 1$ **then**
  $\mathcal{S}^+ = \cap_{k=1}^{j} \mathcal{F}_k^+$
  $\mathcal{S}^- = \cap_{k=1}^{j} \mathcal{F}_k^-$
end

Algorithm 2: The computation of skeptical consequences.

$\mathcal{P} = \{p \leftarrow q, s \leftarrow t\}$ where $\mathcal{A}_\mathcal{P}$ is $\{q \leftarrow \top, t \leftarrow \top\}$ and $\mathcal{O} = \{p\}$, the only explanation is $\mathcal{E} = \{q \leftarrow \top\}$. Additionally to p and q, we conclude that $\neg s$ and $\neg t$ follow skeptically. Without the restriction to minimal explanations, $\mathcal{E} = \{q \leftarrow \top, t \leftarrow \top\}$ would yet be another explanation for $\mathcal{O}$, and consequently only p and q follow skeptically.

3 Computing Skeptical Abduction

In this section we explain how skeptical abduction can be computed. For this purpose, we first need to determine the set of minimal explanations for $\mathcal{P}$, $\mathcal{IC}$ and $\mathcal{O}$. After that, we can identify which literals follow skeptically from $\mathcal{O}$ and $\mathcal{P}$, i.e., which literals follow from all minimal explanations for $\mathcal{O}$.

3.1 Algorithm 1

We define an order on the *candidate explanations* $\mathcal{C}$, i.e., all subsets of $\mathcal{A}_\mathcal{P}$ which are possibly minimal explanations for $\mathcal{O}$. Let $Pow(\mathcal{A}_\mathcal{P})$ be the power set of $\mathcal{A}_\mathcal{P}$ and $\prec$ be any total ordering over $Pow(\mathcal{A}_\mathcal{P})$ s.t. for every $\mathcal{C}, \mathcal{C}' \in Pow(\mathcal{A}_\mathcal{P})$ we have that $|\mathcal{C}| < |\mathcal{C}'|$ implies $\mathcal{C} \prec \mathcal{C}'$. $Ord(\mathcal{A}_\mathcal{P}) = \{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_n\}$ is an ordered set with exactly the elements of $Pow(\mathcal{A}_\mathcal{P})$ s.t. $\mathcal{C}_1 \prec \mathcal{C}_2 \prec \dots \prec \mathcal{C}_n$. All minimal explanations are determined by Algorithm 1.

Lemma 1. *Consider* $\langle \mathcal{P}, \mathcal{A}_\mathcal{P}, \mathcal{IC}, \models^{\mathsf{lfp}} \rangle$ *and observation* $\mathcal{O}$. *The following holds for* $\mathcal{C}_i \in Ord(\mathcal{A}_\mathcal{P})$, $1 \leq i \leq n$: *Whenever* $\mathcal{C}_i$ *is tested as part of condition 1) of Algorithm 1,* $\mathcal{M}_\mathcal{E}$ *contains all minimal explanations* $\mathcal{C}'$ *for* $\mathcal{O}$ *with* $|\mathcal{C}'| < |\mathcal{C}_i|$.

Proof. If there exists a minimal explanation $\mathcal{E} \subset \mathcal{C}_i$ then $\mathcal{E}$ was tested before $\mathcal{C}_i$, because all $\mathcal{E}$ have been tested in order of minimality. As $\mathcal{E}$ is a minimal explanation, it must have been added to $\mathcal{M}_\mathcal{E}$, therefore there cannot exist a minimal explanation smaller than $\mathcal{C}_i$ that is not in $\mathcal{M}_\mathcal{E}$. □

Proposition 1. *Consider* $\langle \mathcal{P}, \mathcal{A}_\mathcal{P}, \mathcal{IC}, \models^{\mathsf{lfp}} \rangle$ *and observation* $\mathcal{O}$. *Let* $\mathcal{M}_\mathcal{E}$ *be the set computed by Algorithm 1.*

1. $\mathcal{M}_\mathcal{E}$ *contains only minimal explanations for* $\mathcal{O}$.
2. *All minimal explanations for* $\mathcal{O}$ *are contained in* $\mathcal{M}_\mathcal{E}$.

Proof. 1. Every $\mathcal{E} \in \mathcal{M}_\mathcal{E}$ satisfies conditions 1) and 2) of Algorithm 1. By Lemma 1 and condition 1) there is no $\mathcal{C} \in Ord(\mathcal{A}_\mathcal{P})$ with $\mathcal{C} \subset \mathcal{E}$ s.t. $\mathcal{C}$ is a minimal explanation for $\mathcal{O}$. By condition 2), as $\mathcal{E}$ explains $\mathcal{O}$, all $\mathcal{E} \in \mathcal{M}_\mathcal{E}$ are minimal explanations for $\mathcal{O}$.
2. (by contradiction) Assume that all possible explanations have been tested and $\mathcal{E} \notin \mathcal{M}_\mathcal{E}$ is a minimal explanation. Then there exists no explanation $\mathcal{E}' \subset \mathcal{E}$, i.e., $\mathcal{E}$ satisfies condition 1), and because $\mathcal{E}$ explains $\mathcal{O}$, it must have been added to $\mathcal{M}_\mathcal{E}$ by condition 2) of Algorithm 1. Therefore, $\mathcal{E} \in \mathcal{M}$. □

3.2 Algorithm 2

Algorithm 2 determines which literals follow skeptically by verifying if they are entailed by all minimal explanations.

Proposition 2. *Consider* $\langle \mathcal{P}, \mathcal{A}_\mathcal{P}, \mathcal{IC}, \models^{\mathsf{lfp}} \rangle$ *and observation* $\mathcal{O}$. *Let* $\mathcal{S}^+, \mathcal{S}^-$ *be computed by Algorithm 2.*

1. $\mathcal{S}^+$ *contains all skeptically entailed positive literals.*
2. $\mathcal{S}^-$ *contains all skeptically entailed negative literals.*

Proof. 1. If $j = 0$ then $\mathcal{O}$ can not be explained, and nothing follows skeptically. Accordingly, the set $\mathcal{S}^+$ will be empty. If at least one minimal explanation exists, we show that all elements in $\mathcal{S}^+$ follow skeptically. Let $A \in \mathcal{S}^+$. Then, because $\mathcal{S}^+ = \cap_{k=1}^{j} \mathcal{F}_k^+$ it follows that $A \in \mathcal{F}_k^+$ with $1 \leq k \leq j$. As Algorithm 2 creates a set $\mathcal{F}_k^+$ for each $\mathcal{E}_k \in \mathcal{M}_\mathcal{E}$, we have that $A \in \mathcal{F}_k^+$ implies $\mathcal{P} \cup \mathcal{E}_k \models^{\mathsf{lfp}} A$, for all $\mathcal{E}_k \in \mathcal{M}_\mathcal{E}$. By Proposition 1, the set $\mathcal{M}_\mathcal{E}$ contains exactly all minimal explanations for $\mathcal{O}$. Consequently, A follows skeptically.
2. can be shown similarly to 1. □

4 A Connectionist Realization

We encode Algorithms 1 and 2 into programs and translate the programs into neural networks with the help of the CORE method. The resulting networks compute $\mathcal{M}_\mathcal{E}$, $\mathcal{S}^+$ and $\mathcal{S}^-$.

4.1 The CORE Method

The CORE method [8; 2] translates programs into neural networks. It is based on the idea that feed-forward networks can approximate almost any function arbitrarily well, and that the semantics of a logic program is captured by the $T_\mathcal{P}$ operator.

We represent propositional variables by natural numbers; let $n \in \mathbb{N}$ be the largest number occurring in $\mathcal{P}$. The network associated with $\mathcal{P}$ is constructed as follows:

[2] This set is computed by Algorithm 1.

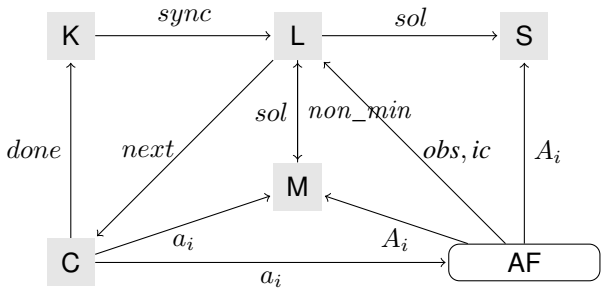

Figure 1: Overview of the network: Components and the connections between them.

1. The input and output layer is a vector of binary threshold units (with threshold .5) of length n, where the ith unit in each layer represents the variable i, $1 \leq i \leq n$. Additionally, each unit in the output layer is connected to the corresponding unit in the input layer with weight 1.
2. For every clause of the form $A \leftarrow L_1 \wedge \cdots \wedge L_k$ with $k \geq 0$ occurring in $\mathcal{P}$ do:
   (a) Add a binary threshold unit c to the hidden layer.
   (b) Connect c to the unit representing A in the output layer with weight 1.
   (c) For each L occurring in $L_1 \wedge \cdots \wedge L_k$ connect the unit representing L in the input layer to c. If L is an atom, then set the weight to 1, otherwise to -1.
   (d) Set threshold θ_c of c to $l - 0.5$, where l is the number of positive literals occurring in $L_1 \wedge \cdots \wedge L_k$.

Lemma 2 ([8])**.** *For each program $\mathcal{P}$ there exists a 3-layer recurrent network that computes $T_\mathcal{P}$.*

Lemma 3 ([9])**.** *For each acceptable program $\mathcal{P}$ there exists a 3-layer recurrent network such that each computation starting with an arbitrary initial input converges and yields the unique fixed point of $T_\mathcal{P}$.*

4.2 The Network

[3] provides a connectionist model to compute credulous abduction. We modify their approach such that we only consider minimal explanations, which allows us to compute skeptical solutions. Figure 1 gives an overview of the network. Each component is expressed as a logic program and translated into a neural network using the CORE method. AF is the abductive framework in consideration where we test all minimal explanations. A *counter* C generates all candidates in order of minimality. M tests only the minimal candidates and records which of them are the (minimal) explanations. *Solution* S keeps track of all literals that follow from each minimal explanation and generates the sets $\mathcal{S}^+$ and $\mathcal{S}^-$. Ultimately, these are the sets we are interested in. Whenever a valid solution has been found, *control logic* L informs S and M. Additionally, L requests C to output the next set of abducibles. Finally the clock K snychronizes the system. It instructs L when to evaluate the current solution.

The Abductive Framework AF

In order to detect which $\mathcal{E} \subseteq \mathcal{A}_\mathcal{P}$ explains $\mathcal{O}$ and satisfies $\mathcal{IC}$, we construct $\mathcal{P}' = \mathcal{P} \cup \mathcal{O}' \cup \mathcal{IC}'$ where, given that $\mathcal{O} = \{L_1, \ldots, L_n\}$, $\mathcal{O}' = \{obs \leftarrow L_1 \wedge \cdots \wedge L_n\}$ and $\mathcal{IC}' = \{ic \leftarrow body \mid \bot \leftarrow body \in \mathcal{IC}\}$. Lemma 4 follows immediately from the definitions.

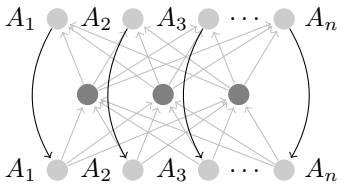

Figure 2: The network for AF, where $\{A_1, \ldots, A_n\} = \mathsf{atoms}(\mathcal{P} \cup \mathcal{O}' \cup \mathcal{IC}')$.

Lemma 4. *Given $\langle \mathcal{P}, \mathcal{A}_\mathcal{P}, \mathcal{IC}, \models^{\mathsf{lfp}} \rangle$ and let $\mathcal{O}$ be a finite set of literals. If $\mathcal{E} \subseteq \mathcal{A}_\mathcal{P}$ explains $\mathcal{O}$ given $\mathcal{P}$ and $\mathcal{IC}$ then $\mathcal{P}' \cup \mathcal{E} \models^{\mathsf{lfp}}$ obs and $\mathcal{P}' \cup \mathcal{E} \models^{\mathsf{lfp}} \neg ic$.*

We can construct the associated network N_{AF} using the CORE method as shown in Figure 2. From this construction and Lemma 3 follows:

Lemma 5. *Given N_{AF} and let $\mathcal{E} \in \mathcal{M}_\mathcal{E}$ be a minimal explanation for $\mathcal{O}$ given $\mathcal{P}$ and $\mathcal{IC}$. If N_{AF} reaches a stable state, then for each neuron i in the output layer of N_{AF} we find: If i is active, then $\mathcal{P}' \cup \mathcal{E} \models^{\mathsf{lfp}} i$. If i is inactive, then $\mathcal{P}' \cup \mathcal{E} \models^{\mathsf{lfp}} \neg i$.*

The Counter C

C outputs all possible combinations of abducibles in the order of minimality. It is divided into the two components C_1 and C_2. C_1 computes all the combinations of k elements out of n and C_2 increases k. This way we first compute all the sets $\mathcal{C} \in Pow(\mathcal{A}_\mathcal{P})$ with $|\mathcal{C}| = 1$. Thereafter, we compute all sets with $|\mathcal{C}| = 2$, $|\mathcal{C}| = 3$,...,$|\mathcal{C}| = n$, where $n = |\mathcal{A}_\mathcal{P}|$.

C_1 starts with a number k of active bits, and is based on two simple rules over a binary vector of length n. Each time the signal δ is activated:

1. The left-most bit b is shifted to the left.
2. When a bit b can not be shifted anymore the last active bit $a < b$ is shifted to the left, all active bits $c > a$ are reset relative to a, and the process restarts from 1.

Eventually all k active bits will be shifted to the left, and C_1 asks C_2 to update the initial configuration of $k+1$ active bits, and starts again. The process ends when C_1 runs with the initial configuration of size $k = n$.

In condition 2. we reset the position of the bits. We need to know which of them were initially active. In order to keep track of the initial k bits, each one is represented by the symbol a_{ij} where i indicates its original and j its current position.

$$\mathsf{C}_1 = \bigcup_{i,j=1}^{n} \left\{ \begin{array}{lll} (1)\ a_{i(j+1)} & \leftarrow & a_{ij}, s_i, \delta \\ (2)\ m_{i-1} & \leftarrow & a_{in}, s_i, \delta \\ (3)\ m_{i-1} & \leftarrow & a_{ij}, a_{(i+1)(j+1)}, m_i \\ (4)\ r_i & \leftarrow & a_{ij}, \neg a_{(i+1)(j+1)}, m_i \\ (5)\ a_{i(j+1)} & \leftarrow & a_{ij}, \neg a_{(i+1)(j+1)}, m_i \\ (6)\ a_{k(j+k-i)} & \leftarrow & r_i, a_{ij}, b_k \text{ (for k>i)} \\ (7)\ a_{ij} & \leftarrow & a_{ij}, \neg a_{i(j+1)}, \neg r_k, \neg v \text{ (for k>i)} \\ (8)\ v & \leftarrow & m_0 \end{array} \right\}$$

$$\mathsf{C}_2 = \bigcup_{i,j=1}^{n} \left\{ \begin{array}{llll} (9) & b_1 & \leftarrow & \\ (10) & b_i & \leftarrow & b_i \\ (11) & b_{i+1} & \leftarrow & b_i, v \\ (12) & done & \leftarrow & b_n, v \\ (13) & \psi & \leftarrow & \neg\psi, next \\ (14) & \delta & \leftarrow & \psi, next \\ (15) & a_{ii} & \leftarrow & \neg\psi, b_i, next \\ (16) & \psi & \leftarrow & \psi, \neg v \\ (17) & s_{i+1} & \leftarrow & s_i, v \\ (18) & s_i & \leftarrow & s_i, \neg s_{i+1} \\ (19) & s_1 & \leftarrow & \neg b_2, \neg\psi, next \\ (20) & a_j & \leftarrow & a_{ij} \end{array} \right\}$$

Due to space restrictions, we only detail the most relevant features of the counter. For each component C_1 and C_2 a correspondent signal δ and ψ is used, δ is used by C_1 to shift the bits to the left, and ψ is used by C_2 to update the initial configuration. They both depend on the $next$ signal. The output of the counter is represented by each neuron a_j.

Proposition 3. *Consider $\mathcal{A}_\mathcal{P}$ and the counter C. The output of C represented by all active atoms a_j generates $Ord(\mathcal{A}_\mathcal{P})$.*

The Clock K

K synchronizes the system. Each cycle of the clock activates the $sync$ signal, which checks if the current solution is valid (sol) and activates the $next$ signal. This causes C to output the next combination of abducibles. The clock has $D+1$ atoms k_i. The value of $D = \frac{1}{2}(P+N+5)$ is chosen to give the rest of the network enough time to stabilize, where P is the longest directed path without repeated atoms in the dependency graph of $\mathcal{P}$, and $N = |\mathcal{A}_\mathcal{P}|$.

In K each atom k_i depends on its predecessor and the additional $\neg non\_min$ condition is used to reset the clock if a non-minimal solution is discarded by component M. The combination $k_0, \neg k_1$ is reached once per cycle and is used to activate the neuron $sync$, which remains active for two consecutive time steps.

$$\mathsf{K} = \bigcup_{i=1}^{D} \{k_i \leftarrow k_{i-1}, \neg non\_min\} \cup \left\{ \begin{array}{ll} k_0 & \leftarrow \neg done, \neg k_n \\ sync & \leftarrow k_0, \neg k_1 \end{array} \right\}$$

This process continues until neuron $done$ is activated (set by (12) on C_2) and prevents k_0 to change its value.

The Control L

L verifies whether *ic* is violated (1) or *obs* is not met (2).

$$\mathsf{L} = \left\{ \begin{array}{llll} (1) & nogood & \leftarrow & ic \\ (2) & nogood & \leftarrow & \neg obs \\ (3) & sol & \leftarrow & sync, \neg nogood \\ (4) & done & \leftarrow & done \\ (5) & next & \leftarrow & sync \end{array} \right\}$$

If this is the case, the current solution is not valid ($nogood$). Both conditions are checked when $sync$ is activated by K. If the conditions are met, an explanation has been found, and sol is activated by (3). (4) serves as a memory for neuron $done$ which indicates thal all explanations have been explored. (5) induces C to output the next possible explanation by activating $next$, which depends on $sync$.

Lemma 6. *Neuron sol is active only when an explanation $\mathcal{E}$ for $\mathcal{O}$ given $\mathcal{P}$ and $\mathcal{IC}$ has been found. It is active for only two time steps.*

Proof. From Lemma 4, if $\mathcal{E}$ explains $\mathcal{O}$, then neuron *obs* will be active, and neuron *ic* inactive, thus $nogood$ can not be activated. Under this conditions when $sync$ is activated by K to check if the current solution is valid, sol will be activated. □

The Minimal Candidates M

M has the tasks to filter C's output s.t. only minimal candidates are tested and to record each minimal explanation that has been found. Given a candidate $\mathcal{C}$, a minimal explanation $\mathcal{E}$ and the set $\mathcal{A}_\mathcal{P}$, we construct the set $\mathcal{Q}_{\mathcal{EC}} = \{q_1, \ldots, q_n\}$ in order to determine $\mathcal{E} \subset \mathcal{C}$, where $q_i = \bot$ iff $a_i \in \mathcal{E}$ and $a_i \notin \mathcal{C}$, and $q_i = \top$ otherwise.
Intuitively, in case all elements of $\mathcal{E}$ are also in $\mathcal{C}$, all $q_i \in \mathcal{Q}_{\mathcal{EC}}$ will be $\top$ indicating that $\mathcal{C}$ is a superset of $\mathcal{E}$ and thus $\mathcal{C}$ can not be minimal (non_min). The process is encoded by adding the following clauses to M, where we label $a_i \in \mathcal{E}$ as $a_i^\mathcal{E}$ in order to distinguish the elements of $\mathcal{E}$ from those of $\mathcal{C}$. For each $\mathcal{E} \in \mathcal{M}_\mathcal{E}$ add to M:

$$\left\{ \begin{array}{l} (1)\ q_i \leftarrow \neg a_i^\mathcal{E} \\ (2)\ q_i \leftarrow a_i, a_i^\mathcal{E} \end{array} \right\} \cup \left\{ \begin{array}{ll} (3)\ non\_min & \leftarrow q_1, \ldots, q_n \\ (4)\ next & \leftarrow non\_min \end{array} \right\}$$

Lemma 7. *Given $\langle \mathcal{P}, \mathcal{A}_\mathcal{P}, \mathcal{IC}, \models^{\mathsf{lfp}} \rangle$ and let $\mathcal{E}$ be a minimal explanation for $\mathcal{O}$ given $\mathcal{P}$ and $\mathcal{IC}$. For any candidate $\mathcal{C}$ where $\mathcal{E} \subset \mathcal{C}$, it holds that $q_i = \top$ for all $q_i \in \mathcal{Q}_{\mathcal{EC}}$.*

Proof. Assume $\mathcal{E} \subset \mathcal{C}$ but $q_i = \bot$ for some i. Then clauses (1) and (2) in M are not satisfied, only if for the corresponding a_i we find both: $a_i \in \mathcal{E}$ and $a_i \notin \mathcal{C}$. But, since $a_i \in \mathcal{E}$ and $\mathcal{E} \subset \mathcal{C}$ it follows that $a_i \in \mathcal{C}$, thus q_i must be true. □

Lemma 8. *Let $\mathcal{C}$ be the current candidate represented by each neuron a_i from the output of the counter C. Given M, the non_min neuron will be active iff $\mathcal{C} \supset \mathcal{E}$ for some $\mathcal{E} \in \mathcal{M}_\mathcal{E}$.*

Proof. Follows from Lemma 7 and Proposition 1. □

Since non_min detects if $\mathcal{E} \subset \mathcal{C}$ for some $\mathcal{E} \in \mathcal{M}$, clause (4) activates the $next$ neuron discarding $\mathcal{C}$ and instructing C to generate the next candidate, thus only minimal explanations will be tested.

To construct the set $\mathcal{M}$ we need to record each minimal explanation found. We provide an upper bound for the number of minimal explanations that can exist. The proof of the following lemma is quite extensive and can be found in [19].

Lemma 9. *Given a set of abducibles $\mathcal{A}_\mathcal{P}$, let $n = |\mathcal{A}_\mathcal{P}|$. Then the maximal number of possible minimal explanations is given by $\max = \frac{(n!)}{\frac{n}{2}!\frac{n}{2}!}$, which is the maximum number of subsets of the same size ($\frac{n}{2}$) out of n elements.*

The following rules are added to M to keep track of how many explanations $\mathcal{E}$ have been found.

$$\left\{ \begin{array}{lll} \mathcal{E}_1 & \leftarrow & sol, \neg\mathcal{E}_1 \\ \mathcal{E}_2 & \leftarrow & sol, \neg\mathcal{E}_2, \mathcal{E}_1 \\ \mathcal{E}_3 & \leftarrow & sol, \neg\mathcal{E}_3, \mathcal{E}_2 \\ \ldots & & \\ \mathcal{E}_{\max} & \leftarrow & sol, \neg\mathcal{E}_{\max}, \mathcal{E}_{\max-1} \end{array} \right\} \cup \{\ \mathcal{E}_i \leftarrow \mathcal{E}_i\ \}$$

When the ith solution has been found, the correspondent $\mathcal{E}_i$ neuron will be active. Because sol is active for only two time steps, it ensures that only one of the rules above will be fired by each activation of sol, and only one minimal explanation will be recorded at each time.
As soon as an $\mathcal{E}_i$ has been found, its elements are recorded. For this purpose we introduce the symbol $a_i^{\mathcal{E}_j}$ which states that atom a_i belongs to $\mathcal{E}_j$. Then, for each $a_i \in \mathcal{A}$ and each $\mathcal{E}_j$ with $1 \leq j \leq \max$, we add the following rules to M :

$$\left\{\begin{array}{l}(1)\ a_i^{\mathcal{E}_j} \leftarrow a_i, \mathcal{E}_j, \neg block\_\mathcal{E}_j \\ (2)\ a_i^{\mathcal{E}_j} \leftarrow a_i^{\mathcal{E}_j}\end{array}\right\} \cup \{(3)\ block\_\mathcal{E}_i \leftarrow \mathcal{E}_i\ \}$$

(1) is fired as soon as $\mathcal{E}_j$ is activated, and records its elements $a_i^{\mathcal{E}_j}$. The additional condition $\neg block\_\mathcal{E}_j$ ensures that we record the values only when the $\mathcal{E}_j$ solution was found. (2) serves as a memory for each $a_i^{\mathcal{E}_j} \in \mathcal{E}_j$. (3) blocks the current explanation; thus, only values present at the moment $\mathcal{E}_j$ was detected are recorded.

Lemma 10. *Consider $\langle \mathcal{P}, \mathcal{A}_\mathcal{P}, \mathcal{IC}, \models^{\mathsf{lfp}} \rangle$, observation $\mathcal{O}$ and the components AF, C, K, L and M. At the moment sol neuron is active, a minimal explanation $\mathcal{E}$ for $\mathcal{O}$ is found. Then for each active neuron A in the output layer of AF it holds: $\mathcal{P} \cup \mathcal{E} \models^{\mathsf{lfp}} A$ and, for each inactive neuron A in the output layer of AF it holds $\mathcal{P} \cup \mathcal{E} \models^{\mathsf{lfp}} \neg A$.*

Proof. From Lemma 3 we know that N_{AF} computes $T_\mathcal{P}^{\mathsf{lfp}}$. From Lemmas 6 and 10 we conclude that each explanation detected by the signal sol is minimal. □

The Solution S

Component S allows us to obtain the set $\mathcal{S}^+$ by computing the intersection of each set $\mathcal{F}_j^+ = \{A \mid \mathcal{P} \cup \mathcal{E}_j \models^{\mathsf{lfp}} A\}$. Instead of first obtaining all sets $\mathcal{F}_j^+$ and then computing $\mathcal{S}^+$, the following logic program progressively constructs $\mathcal{S}^+$ by intersecting the current set $\mathcal{F}_j^+$ with the last computed value of $\mathcal{S}^+$. Thus, we compute $\mathcal{S}_j^+ = \mathcal{F}_j^+ \cap \mathcal{S}_{j-1}^+$. This avoids introducing a memory to remember each set $\mathcal{F}_j^+$ and when all k minimal explanations are found the set $\mathcal{S}_k^+$ will correspond to the set $\mathcal{S}^+$. For the case $j = 1$, we define $\mathcal{S}_1^+ = \mathcal{F}_1^+$. A similar construction yields $\mathcal{S}_k^-$ which corresponds to $\mathcal{S}^-$.

Lemma 11. *If all minimal explanations have been found, the sets $\mathcal{S}_k^+$ and $\mathcal{S}_k^-$ correspond to the sets $\mathcal{S}^+$ and $\mathcal{S}^-$, resp.*

Proof. For $j = 1$ is trivial. In case $j > 1$, $\mathcal{S}_j^+ = \mathcal{F}_j^+ \cap \mathcal{S}_{j-1}^+$ and since $\mathcal{S}_{j-1}^+ = \cap_1^{j-1} \mathcal{F}_i^+$ it follows that $\mathcal{S}_j^+ = \mathcal{F}_j^+ \cap \mathcal{F}_{j-1}^+ \cap \mathcal{F}_{j-2}^+ \cap \cdots \cap \mathcal{F}_1^+$, thus when the final solution k is found, $\mathcal{S}_k^+ = \mathcal{S}^+$. The proof is similar for $\mathcal{S}^-$. □

In order to compute $\mathcal{S}^+$, we first need to construct $\mathcal{F}_1^+$. This is achieved by introducing a copy A_i^+ for each $A_i \in \mathcal{R}$ such that $A_i^+ = \top$ iff $\mathcal{P} \cup \mathcal{E}_1 \models^{\mathsf{lfp}} A_i$, with $\mathcal{E}_1 \in \mathcal{M}_\mathcal{E}$. For each $A_i \in \mathcal{R}$ we add the following clauses to S:

$$\left\{\begin{array}{ll}(1) & A_i^+ \leftarrow A_i, sol \\ (2) & A_i^+ \leftarrow A_i^+\end{array}\right\}$$

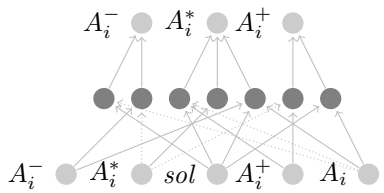

Figure 3: Network for S

Consider (1): By Lemma 6 and 10, sol is active only when a minimal explanation $\mathcal{E}$ for $\mathcal{O}$ is found and for each active neuron A_i in the output layer of AF, $\mathcal{P} \cup \mathcal{E} \models^{\mathsf{lfp}} A_i$. (1) ensures that A_i^+ is active iff $\mathcal{P} \cup \mathcal{E} \models^{\mathsf{lfp}} A_i$, and in particular $A_i^+ = \top$ iff $\mathcal{P} \cup \mathcal{E}_1 \models^{\mathsf{lfp}} A_i$. Additionally, by (2), we remember that $A_i^+ \in \mathcal{F}_j^+$ for some j, thus we introduce a recurrent connection for A_i^+.
Besides remembering when an atom was present in a solution $\mathcal{E}$ we need to detect if this atom does not occur in all other solutions. Atoms which follow credulously but not skeptically are labelled by A^*. Let A_i be the atom corresponding to an active neuron A_i^+. $A_i \notin \mathcal{S}^+$ if $A_i \notin \mathcal{F}_j^+$ for some j. This is detected by (3) and the atom A_i^* is activated:

$$\left\{\begin{array}{ll}(3) & A_i^* \leftarrow A_i^+, \neg A_i, sol \\ (4) & A_i^* \leftarrow A_i^*\end{array}\right\}$$

If for the current explanation $\mathcal{E}_j$ it holds $\mathcal{P} \cup \mathcal{E}_j \models^{\mathsf{lfp}} \neg A_i$ and, $\mathcal{P} \cup \mathcal{E}_k \models^{\mathsf{lfp}} A_i$ for any $k < j$ then $A_i \notin \mathcal{S}^+$ and, by (3), the atom A_i is labeled A_i^*. Finally, regardless of any further result, by (4), an atom A_i^* is always left out of $\mathcal{S}^+$. Thus, we add a recursive connection to remember its state.
Finally, S detects only the positive literals that skeptically follow from $\mathcal{P}$ and $\mathcal{O}$. To obtain the corresponding negative literals we extend S by the following clauses:

$$\left\{\begin{array}{ll}(5) & A_i^- \leftarrow \neg A_i, sol \\ (6) & A_i^- \leftarrow A_i^- \\ (7) & A_i^* \leftarrow A_i^-, A_i, sol\end{array}\right\}$$

The corresponding network is shown in Figure 3.

Claim 1. *Consider $\langle \mathcal{P}, \mathcal{A}_\mathcal{P}, \mathcal{IC}, \models^{\mathsf{lfp}} \rangle$, observation $\mathcal{O}$ and the components AF, C, K, LM and S. If neuron sol is active, then $\mathcal{S}_j^+ = \{A_i \mid A_i^+ = \top\} \setminus \{A_i \mid A_i^* = \top\}$ and $\mathcal{S}_j^- = \{\neg A_i \mid A_i^- = \top\} \setminus \{A_i \mid A_i^* = \top\}$.*

The resulting network

Using the CORE method we translate $\mathcal{P}$ into a neural network N_{AF} and we add recurrent connections from its output to its input layer. The resulting network computes $T_\mathcal{P}^{\mathsf{lfp}}$, given that $\mathcal{P}$ is a acceptable.

The networks N_{C}, N_{L}, N_{K}, N_{M}, N_{S} corresponding to the components C, L, K, M, S, resp., are obtained by translating the theory representing each component using the CORE method. Each resulting sub-network has the property that it computes the associated $T_\mathcal{P}$ operator, thus the semantics detailed above are preserved. In the following, let $N = \langle N_{\mathsf{AF}}, N_{\mathsf{C}}, N_{\mathsf{L}}, N_{\mathsf{K}}, N_{\mathsf{M}}, N_{\mathsf{S}} \rangle$.

Claim 2. *Let* $\mathcal{R}^+ = \{A_i \mid A_i^+ = \top\}$, $\mathcal{R}^- = \{A_i \mid A_i^- = \top\}$, *and* $\mathcal{R}^{bad} = \{A_i \mid A_i^* = \top\}$. *If* N *reaches a stable state, then for all minimal explanations* $\mathcal{E} \in \mathcal{M}_\mathcal{E}$ *and for each atom* $A_i \in (\mathcal{R}^+ \setminus \mathcal{R}^{bad})$ *we find* $\mathcal{P} \cup \mathcal{E} \models^{\mathsf{lfp}} A_i$*; likewise, for each atom* $A_i \in (\mathcal{R}^- \setminus \mathcal{R}^{bad})$ *we find* $\mathcal{P} \cup \mathcal{E} \models^{\mathsf{lfp}} \neg A_i$.

Claim 3. *If* N *reaches a stable state, then* $\mathcal{S}^+ = \mathcal{R}^+ \setminus \mathcal{R}^{bad}$ *and* $\mathcal{S}^- = \{\neg A \mid A \in \mathcal{R}^- \setminus \mathcal{R}^{bad}\}$.

Remarks

Positive cycles in the framework AF can be problematic as they may introduce a memory in the network. To handle them properly the AF component has to be relaxed at each $next$ signal. This can be done by connecting the neuron $next$ to each neuron representing a rule in $\mathcal{P}$ in the input layer of AF, and erase possible memories introduced by such cycles.

5 Conclusion

We have developed a connectionist network for skeptical reasoning based on the CORE method and the ideas published in [3] for credulous reasoning. The CORE method has already been extended to many-valued logics in [15; 21], but the skeptical reasoning part has not been considered yet. We intend to extend our skeptical reasoning approach to three-valued Łukasiewicz logic to realize our ultimate goal as outlined in the introduction, viz. to develop a connectionist model for human reasoning tasks.

Although the AF always reaches a stable state, other components are represented by programs which are not acceptable. We need to provide the missing proofs for Claims 1, 2 and 3 which can be built showing that the network N terminates, i.e., that it reaches a least fixed point.

References

[1] K. Apt and M. van Emden. Contributions to the theory of logic programming. *J. ACM*, 29:841–862, 1982.

[2] S. Bader and S. Hölldobler. The Core method: Connectionist model generation. In S. Kollias, A. Stafylopatis, W. Duch, and E. Ojaet, eds., *Proc. 16th International Conference on Artificial Neural Networks (ICANN)*, vol. 4132 of *LNCS*, 1–13. Springer, 2006.

[3] A. d'Avila Garcez, D. Gabbay, O. Ray, and J. Woods. Abductive reasoning in neural-symbolic learning systems. *TOPOI*, 26:37–49, 2007.

[4] E.-A. Dietz, S. Hölldobler, and M. Ragni. A computational logic approach to the abstract and the social case of the selection task. In *Proc. 11th International Symposium on Logical Formalizations of Commonsense Reasoning*, 2013.

[5] E.-A. Dietz, S. Hölldobler, and L. M. Pereira. On indicative conditionals. In S. Hölldobler and Y. Liang, eds., *Proc. First International Workshop on Semantic Technologies*, vol. 1339 of *CEUR Workshop Proc.*, 19–30, 2015. `http://ceur-ws.org/Vol-1339/`.

[6] E.-A. Dietz, S. Hölldobler, and M. Ragni. A computational logic approach to the suppression task. In N. Miyake, D. Peebles, and R. P. Cooper, eds., *Proc. 34th Annual Conference of the Cognitive Science Society*, 1500–1505. Cognitive Science Society, 2012.

[7] M. Fitting. Metric methods – three examples and a theorem. *J. of Logic Programming*, 21(3):113–127, 1994.

[8] S. Hölldobler and Y. Kalinke. Towards a new massively parallel computational model for logic programming. In *Proc. ECAI94 Workshop on Combining Symbolic and Connectionist Processing*, 68–77. ECCAI, 1994.

[9] S. Hölldobler, Y. Kalinke, and H.-P. Störr. Approximating the semantics of logic programs by recurrent neural networks. *Applied Intelligence*, 11:45–59, 1999.

[10] S. Hölldobler and C. D. P. Kencana Ramli. Contraction properties of a semantic operator for human reasoning. In L. Li and K. K. Yen, eds., *Proc. Fifth International Conference on Information*, 228–231. International Information Institute, Tokyo, 2009.

[11] S. Hölldobler and C. D. P. Kencana Ramli. Logic programs under three-valued Łukasiewicz's semantics. In P. M. Hill and D. S. Warren, eds., *Logic Programming*, vol. 5649 of *LNCS*, 464–478. Springer, 2009.

[12] S. Hölldobler, T. Philipp, and C. Wernhard. An abductive model for human reasoning. In *Proc. 10th International Symposium on Logical Formalizations of Commonsense Reasoning*, 2011.

[13] S. Hölldober. *Logik und Logikprogrammierung: Grundlagen*. Synchron Publishers GmbH, Heidelberg, 2009.

[14] A. C. Kakas, R. A. Kowalski, and F. Toni. Abductive Logic Programming. *J. Logic and Computation*, 2(6):719–770, 1993.

[15] Y. Kalinke. Ein massiv paralleles Berechnungsmodell für normale logische Programme. Master's thesis, TU Dresden, Fakultät Informatik, 1994. (in German).

[16] A.C. Kakas and P. Mancarella. Generalized Stable Models: a Semantics for Abduction. In *Proc. 9th European Conference on Artificial Intelligence*, 385–391. Pitman, 1990.

[17] J. W. Lloyd. *Foundations of Logic Programming*. Springer, 1984.

[18] J. Łukasiewicz. O logice trójwartościowej. *Ruch Filozoficzny*, 5:169–171, 1920. English: On Three-Valued Logic. In: *Jan Łukasiewicz Selected Works*. (L. Borkowski, ed.), North Holland, 87-88, 1990.

[19] L. Palacios. A connectionist model for skeptical abduction. Project thesis, TU Dresden, Fakultät Informatik.

[20] O. Ray and A. d'Avila Garcez. Towards the integration of abduction and induction in artificial neural networks. In *Proc. ECAI'06 Workshop on Neural- Symbolic Learning and Reasoning*, 41–46, 2006.

[21] A. Seda and M. Lane. Some aspects of the integration of connectionist and logic-based systems. In *Proc. Third International Conference on Information*, 297–300, International Information Institute, Tokyo, 2004.

[22] K. Stenning and M. van Lambalgen. *Human Reasoning and Cognitive Science*. MIT Press, 2008.

Towards an Artificially Intelligent System: Possibilities of General Evaluation of Hybrid Paradigm

Ondřej Vadinský
Department of Information and Knowledge Engineering
University of Economics, Prague, Czech Republic
ondrej.vadinsky@vse.cz

Abstract

The paper summarizes presumptions of intelligence from the point of view of the artificial general intelligence and with focus on the hybrid paradigm. Particularly, the paper concerns with possible ways in which Universal Intelligence definition and the derived Algorithmic Intelligence Quotient test can be used to evaluate a hybrid system, the hybrid paradigm and the paradigms of artificial intelligence. A preliminary search suggests that Algorithmic Intelligence Quotient test can be used for practical evaluation of systems on condition that their interfacing is technically solved, while the Universal Intelligence definition can be used for formal analysis in case a suitable formal description of the system or paradigm is devised. The long-standing issue of evaluating general artificial intelligence albeit crucial has been neglected. However, it can help focusing research efforts, as well as answering some questions regarding the intelligence itself.

1 Introduction

Recently, the original question of artificial intelligence (AI): "What is intelligence?" has again been brought into focus, cf. [Turing, 1950; Legg and Hutter, 2007]. It seems, that the question is, indeed, crucial for reaching the original goal of AI by some dubbed as strong AI or in more current terms artificial general intelligence (AGI), cf. [Searle, 1980; Goertzel, 2014]. As pointed out by Vadinský [2013] and detailed later in [2014], it is a multidisciplinary endeavor reaching to various areas of philosophy and other cognitive sciences, not only a limited AI problem. This is also supported by ongoing research of cognitive architectures, see e.g. [Sun, 2007]. An outline of these issues will be given in Section 2.

Throughout the history of AI, several paradigms emerged: symbolism, connectionism, hybrid approach and situated cognition, suppling the field with various approaches to reach the goal, see e.g. [Sun, 2001; Walter, 2010]. Of these paradigms the paper concerns mainly the hybrid approach since it synergically combines both symbolism and connectionism. Certain aspects of situated cognition are also discussed. A brief summary of the paradigms of AI will be given in Section 3.

Taking into account various ways in which intelligence was delimited, especially Legg and Hutter's [2007] Universal Intelligence definition, an evaluation of AI and its methods seems feasible. In Section 4, several possibilities of this evaluation will be outlined. These ideas will focus on AGI-point-of-view analysis of a hybrid system in particular (4.1), the hybrid approach (4.2), and the paradigms of AI in general (4.3).

2 Artificial and Natural Intelligence

To grasp intelligence, philosophical reflection will be called to help (2.1). An outline of the knowledge of cognitive science as well as some examples of ongoing research of cognitive architectures will be given to illustrate the need for general-oriented multi-disciplinary approach (2.2). Further, the strong – weak and general – specific AI distinctions will be summarized (2.3). Finally, attempts on defining AI will be described (2.4).

2.1 Philosophical Presumptions of Intelligence

So far the only true intelligence known to us is human intelligence. Therefore, any effort to build AI should somehow relate to human intelligence, although this does not necessarily call for its duplication. Of course, research on animal behavior and of AI itself, can also contribute. This way presumptions of intelligence, that is properties essentially tied with intelligence, can be identified to be later used as a guidance to build an AI.

Philosophical reflections of what is intelligence and thought can be traced back at least to Descartes [1637], who notices universality of thought and its connection to language and rational speech. Similarity to the well-known Turing test [1950], which was at the beginning of AI as a field of study, can be easily seen. Expanded tests of intelligence were later proposed e.g. by Harnad [1991], who focuses on full repertoire of human intelligent behavior, and Schweizer [2012], who accentuates the ability of a species to evolve its intelligent behavior. Searle [1980] in his Chinese Room Argument notices the connection of intelligence with meaning, understanding and intentionality. However, presumptions of intelligence can be understood even more widely, e.g. to the level of consciousness, as discussed in [Dennett, 1993].

2.2 Cognitive Presumptions of Intelligence

As de Mey [1992] points out, information processing requires a system to have a world representation in some kind of model. However, de Mey also stresses the ways in which the model interacts with the world through perception and action. A stratified model of perception shows, how the expectations of a system based on its world model contribute to its perception of some object and how this propagates back to the model. The way in which the model is updated follows according to de Mey two steps: 1) implicit knowledge is formed, 2) further interaction with the object makes the implicit knowledge explicit.

To explain cognition, Sun [2007] proposes to use an integrated hierarchical approach. Various sciences encounter the cognition on different levels of abstraction: social, psychological, componential and physiological. This creates sets of constrains both from the upper levels, as well as from the lower levels of abstraction. Using this approach, cognitive architectures – that is domain-generic computational models capturing essential structures and processes of the mind – can be build. Those are useful both for understanding the cognition as well as for building AI systems. Examples of cognitive architectures include: modular neural-symbolic architecture CLARION [Sun, 2007], modular declarative-procedural architecture ACT-R [Anderson *et al.*, 2004], or recursive algorithmic interpolating model Ouroboros [Thomsen, 2013].

2.3 Strong, Weak, Specific and General AI

The ultimate goal of AI can be understood differently. To illustrate it, a strong – weak distinction made by Searle [1980] can be adopted. Originally, Searle based his distinction on whether AI is the explanation of mind (Strong AI), or only a tool for modeling and simulating mind (Weak AI). He also identifies the strong AI with traditional symbolic approach. However, the distinction can be interpreted more freely: Strong AI would be such an AI system, that it would be comparably powerful to human intelligence, bearing in mind its universality. On the other hand, Weak AI would not be universal, it would only provide humans with useful tools to solve certain problems.

This interpretation of Searle's strong – weak AI distinction is similar to more recent general – specific AI distinction, as given e.g. in [Goertzel, 2014]. Specific AI offers programs capable of solving a specific task, or perhaps a limited set of tasks, no matter how the task itself is sophisticated. General AI strives for programs capable of general problem solving, or general intelligent action. Such an AI would be able to solve a broad set of tasks, ranging from simple to complex.

2.4 Defining Artificial and Natural Intelligence

When dealing with AI, a question: "What is intelligence?" naturally comes to mind. The question was already noted by Turing [1950], however, he sidestepped it. Only recently, the question came back into research focus. The issue can be approached from different angles: 1) focusing more on the testing aspect is so called psychometric AI (PAI) of [Bringsjord and Schimanski, 2003], 2) stressing the need for a definition is the work of [Legg and Hutter, 2007].

The field of psychology called psychometrics deals with a systematical measurement of psychological properties (especially intelligence) in humans using various tests. According to [Bringsjord and Schimanski, 2003] it gives an answer to the question, what is intelligence, therefore AI should be understood as PAI and should focus on "building information-processing entities capable of at least solid performance on all established, validated tests of intelligence and mental ability." This results in an iterative approach of integrating the ability of solving yet another test into the entity in question – which is useful from the engineering perspective.

However as a testing approach, PAI is somewhat impractical, since it deals with all established and validated tests which is an open set. PAI also does not explicitly concern the issue of defining intelligence and leaves it to psychology. That may be considered a benefit by some, however questions can be raised, whether psychological definitions of human intelligence implicitly present in the tests can be directly applied to an artificial system. A more general approach supported by multidisciplinary interaction may be needed. The limitations of PAI are also considered by [Besold *et al.*, 2015] stating several arguments questioning the necessity and sufficiency conditions of directly using human intelligence tests to measure machine intelligence. This also calls for generalization and improvement of tests used by PAI.

As argued by [Legg and Hutter, 2007], to achieve the ultimate goal of AI, intelligence has to be precisely and formally defined. Ideally, the definition should allow for some sort of comparison or measurement to serve as a guide for reaching the goal. To create such a definition, Legg and Hutter studied a broad variety of definitions, theories, and tests of human, animal, and artificial intelligence. Noting their common essential features, Legg and Hutter derived a following informal definition: "Intelligence measures an agent's ability to achieve goals in a wide range of environments." The formalization of the definition dubbed as Universal Intelligence is shown by Figure 1.

$$\Upsilon(\pi) := \sum_{\mu \in E} 2^{-K(\mu)} V_\mu^\pi$$

Figure 1: Universal Intelligence Υ of agent π is given by agent's ability to achieve goals as described by a value function V_μ^π as an expected sum of all future rewards over a set of environments μ weighted by Kolmogorov complexity function K. For more detailed explanation see [Legg and Hutter, 2007].

Looking at the formal definition by [Legg and Hutter, 2007] at Figure 1, following important building blocks can be noted:

- The definition considers environment μ, agent π and their iterated interaction through actions a_i of the agent, its perceptions (observations) o_i and rewards r_i originating in the environment. The environment is described as a computable probability measure μ of perceptions and rewards given current interaction history.
- With all computable environments considered and many

agent's hypotheses of the current environment in the current iteration existing, Kolmogorov complexity $K(\mu)$ is used as a measure in place for Occam's razor $2^{-K(\mu)}$. If a short program can be used to describe the probability measure of an environment (ie. as a hypothesis about the environment), than the environment has a low complexity, since Kolmogorov complexity is based on the length of the shortest program describing a sequence of bits. Therefore, complex environments (hypotheses) are less influential on the agent's overall performance, than the simple ones. However, even complex environments (hypotheses) are considered if consistent with history of interactions.

- Agent's ability to achieve goals is described by a value function: $V_\mu^\pi := \mathrm{E}\left(\sum_{i=1}^{\infty} r_i\right) \leq 1$ basically as maximizing the expected future rewards r_i given past interaction with the environment. Temporal preference is present in the way, how rewards are distributed by the environment. That is, the task or environment itself decides whether a slow-learning but more accurate or fast-learning but inaccurate solution is better.

The definition proposed by Legg and Hutter [2007] enables ordering of performance of various agents, ranging from randomly behaving to theoretically optimally intelligent agent AIXI. As environments are weighted by their complexity and all Turing-computable environments are considered, to achieve a high level of Universal Intelligence, a true generality of the agent is required. As the definition draws from fundamental concepts of computation, information, and complexity, it is not culturally biased or anthropocentric. The definition is by itself not a test, however it has a potential to found a basis for some approximative test of intelligence.

The potential was examined by [Legg and Veness, 2013] proposing an approximation of Universal Intelligence measure called Algorithmic Intelligence Quotient (AIQ) as shown in Figure 2. Legg and Veness took many steps to transform the original definition into a practically feasible test resulting in an open source prototype implementation.

$$\hat{\Upsilon}(\pi) := \frac{1}{N} \sum_{i=1}^{N} \hat{V}_{p_i}^{\pi}$$

Figure 2: AIQ estimate of Universal Intelligence $\hat{\Upsilon}$ of agent π is given by agent's ability to achieve goals as described by an empirical value function $\hat{V}_{p_i}^{\pi}$ as total reward returned from a single trial of an environment program p_i averaged over N randomly sampled environment programs. For more detailed explanation see [Legg and Veness, 2013].

Looking at the formula of AIQ test by [Legg and Veness, 2013] at Figure 2, following differences from the Universal Intelligence definition can be noted:

- The test considers a finite sample of N environment programs p_i, agent π and their interaction. Same environment can be described by several programs and same program can be included in the sample many times.
- With only N environment programs considered, simple average is taken. However, the notion of Occam's razor is kept in the way environment programs are sampled since Solomonoff's Universal Distribution is used: $M_{\mathcal{U}}(x) := \sum_{p:\mathcal{U}(p)=x*} 2^{-l(p)}$. Therefore, a shorter program has a higher probability of being used, but all programs are considered, not only the shortest as is the case with Kolmogorov complexity.
- Empirical value function $\hat{V}_{p_i}^{\pi}$ is used since only a limited number of iterations is tried. Also, the rewards given by the environment program are no longer bound by 1 as is the case with the definition, nor are in any way discounted to specify the temporal preference on this level. That is, total reward returned from a single trial of agent – environment interaction is used.

The AIQ test proposed by [Legg and Veness, 2013] enables testing of agents supplied internally by the prototype implementation or externally via a custom wrapper as is the case with the AIXI approximation. The test is configurable namely by setting size of the environment programs sample, number of iterations for agent – environment interaction, sizes of observation and action space, or computation limit per iteration. The test uses a simple reference Turing machine (a modified BF reference machine). On the choice of reference machine, however, the results are dependent.

3 Paradigms of Artificial Intelligence

As the AI field evolved, several paradigms appeared. Symbolism (3.1) draws its inspiration from logic and abstract thought, while connectionism (3.2) is inspired by biological neural networks. Realizing their complementarity, hybrid paradigm (3.3) tries to reach a synergic combination. Situated cognition (3.4) is inspired by phenomenology and biology and accents interaction of an intelligent system with its environment.

3.1 Symbolic Paradigm

Drawing its inspiration from the very beginnings of computer science, symbolic paradigm is tightly connected to the idea of universal computation of Turing machine and ensuing computational theory of mind and functionalism, cf. [Turing, 1936; Dennett, 1993]. Coined as Physical Symbol System by [Newell and Simon, 1976], the paradigm deals with an explicit representation of knowledge in the form of symbols structured into physical patterns by an external observer. However, the way in which symbols gain their meaning, known as the Grounding Problem, is crucial [Harnad, 1990]. Several solutions have been proposed, which either deal with more symbols of different kinds, as e.g. Rapaport's [1995] Syntactic Semantics, and recent semantic approaches such as semantic web [Gray, 2014], or robotic interaction with the world, as Harand's original solution, or even result in adopting another paradigm. Nevertheless, symbolic paradigm brings good results when algorithm is known, knowledge are explicitly represented and the data processing is mostly sequential.

3.2 Connectionist Paradigm

Inspired by massive parallelism of human brain, connectionism is based on relatively simple computational units connected to a complex network in which higher ability emerges in accordance with emergentist theory of mind, cf [Churchland and Churchland, 1990; Sun, 2001]. Starting with McCulloch and Pitts'[1943] artificial neuron complemented by Hebb's [1949] unsupervised learning, the paradigm evolved more complex models such as Rosenblatt's [1958] perceptron with supervised back-propagation learning of [McClelland *et al.*, 1986]. Connectionist paradigm is suitable for tasks where knowledge is implicit and learning is needed providing parallel and robust solution.

3.3 Hybrid Paradigm

Realizing the complementarity of both symbolic as well as connectionist paradigm, it seems only natural to try a hybrid approach. Inspired by dual-process approaches to theory of mind, hybrid paradigm seeks synergy in dual representation of knowledge, e.g. [Sun *et al.*, 2005]. The architecture of hybrid systems is an important consideration: usually it is some version of a heterogeneous modular system, however several variants of tight or loose coupling of connectionist and symbolic modules exist [Sun, 2001]. There are two ways in which learning of a hybrid system is realized: a bottom – up approach is basically emergence of explicit knowledge from implicit layer, while a top – down approach concerns a gradual descend of explicit knowledge in an assimilative way into the implicit layer [Sun, 2007].

3.4 Situated Cognition

Rooted in French phenomenology of Merleau-Ponty [1942; 1945] and inspired by Maturana and Varela's [1979] autopoietic approach to biology, the paradigm of situated cognition stresses the role of physical body and interaction through perception and action with the environment creating a species-specific Umwelt. Manifested mainly in Brooks' [1991] reactive approach to robotics, the situated cognition suppresses the role of representation to a varying degree. An explanation of different schools in situated cognition, i.e. embodied, embedded, extended, and enacted cognition is given e.g. by Walter [2010].

4 AGI Evaluation and Hybrid Paradigm

Having outlined presumptions of intelligence, a formal definition, and a related practical test of AI, several research possibilities concerning evaluation of AI paradigms and systems arise. The paper will mainly discuss ways of evaluating a chosen hybrid system (4.1), and a hybrid paradigm in general (4.2). Alternatives of evaluating AI paradigms will be also mentioned (4.3), since a hypothesis is held, that hybrid approach is more suitable for reaching AGI.

4.1 Possibilities of Evaluating Presumptions of Intelligence of a Hybrid System

First, a suitable hybrid system should be chosen. Then, there seem to be two ways of evaluating presumptions of intelligence of a hybrid system: A system could be tested using the AIQ test. A system could be formally analyzed using the Universal Intelligence definition. Let us now consider what such possibilities require and what results can be achieved.

1. A choice of a suitable hybrid system:
   - A broad search of existing hybrid systems should be conveyed and some criteria defined.
   - Obviously, to test a system using the AIQ test, the system should be implemented.
   - For a formal analysis using the Universal Intelligence definition it should suffice that the system is formally defined.
2. AIQ test of the chosen hybrid system:
   - Technical issues can be expected when connecting the system with the test. Here, an open source system would be beneficial, as modifications necessary for the test could be realized. As for closed source systems, it may be feasible to develop some kind of a wrapper, if an interface is well defined. The test itself is open source, therefore some modifications needed by a system can be feasible.
   - Obviously, the results will enable a comparison of the evaluated system with other tested systems. So far however, only simple agents and AIXI approximations were tested to the best of my knowledge.
   - The results will allow for incremental testing of improved versions of the system, however it is unclear, if they can shed some light on what to improve specifically.
3. Formal analysis of the chosen hybrid system:
   - As the definition focuses on the interaction of the agent with environments, a formalization of the system focused on the interaction is needed. It is an open question, if there are hybrid systems formally described in such a way.
   - If there are systems formalized structurally, a feasibility of a transformation to the needed form is the issue. As the definition is constructed so that it is applicable on the widest possible range of agents, it tries not to constrain agent's structure, however, some structures may be better than others in the terms of resulting behavior, so perhaps this is also worth of consideration.
   - The analysis can be also done less rigorously. Considering what sorts of tasks (environments) can the system succeed in and how complex the tasks are, can yield some insight of its Universal Intelligence. This can be further specified if some estimates can be made regarding the agent's ability of keeping history of its interactions with the environment and making predictions for the future.
   - The results will, obviously, allow for ordering the systems according to their Universal Intelligence and should help comparing different design improvements. However, this will strongly depend on the precision and suitability of formalization of the analyzed system, as discussed above.

4.2 Possibilities of Evaluating Presumptions of Intelligence of Hybrid Paradigm

The evaluation of presumptions of intelligence can be brought to a more general level – the hybrid paradigm itself. Here, an inductive approach generalizing results of hybrid systems in the AIQ test and in the formal analysis according to the Universal Intelligence definition could be practicable. Also, it may be possible to use a deductive approach based on the Universal Intelligence definition and a general formal description of hybrid systems. Let us now consider requirements and possible results of such approaches

1. Induction from AIQ test results and formal analyses:
    - An obvious condition is the feasibility of a test and an analysis of a hybrid system as discussed in previous section.
    - For an inductive generalization tests of several hybrid systems should be performed. Here, an issue of induction arises: Is it necessary to test all existing hybrid systems for the induction to be relevant? Or will it be reasonably sufficient to test some sample of existing systems? Having a poor result of a chosen hybrid system in AIQ test as a base for generalization, would it indicate, that the hybrid approach is poor, or that the system is yet incomplete?
    - Such cases would support the need to generalize from the widest base possible, ideally somehow taking into the account, that some existing implementations may not be implemented well. Perhaps, taking the complexity of a system into consideration similarly as it is used in the test and the definition to weight environments might help.
    - Results of this approach should give some general standing of the hybrid paradigm regarding the Universal Intelligence and AIQ. By itself it may not be very interesting, however comparison to AIXI, or other evaluated types of agents can be feasible and stimulating.
2. Deduction from formal description of hybrid paradigm:
    - Clearly, a formal definition or description of hybrid paradigm is required. Is it possible to construct it and how will it differ from a formal definition of a specific hybrid system?
    - Moreover, the description has to be in a form suitable for the Universal Intelligence definition, which stresses the ability of an agent to interact with its environment.
    - Therefore a formal description of the structure of hybrid paradigm will not suffice unless it is readily transformable to the form describing the interaction. There are deeper questions behind this issue: Is it possible to predict or describe interaction of the agent with its environment based on the inner structure of the agent? To what degree does the structure contribute to the agent's interaction and performance? Are there some generally better structures?
    - Similar considerations apply for the results as for the previous case. Also, a comparison with the results of the induction can tell us how do the current hybrid systems fulfill the potential of the hybrid paradigm. This also applies for a comparison with an individual system.

4.3 Possibilities of Evaluating Presumptions of Intelligence of the Paradigms of AI

Taking the idea of evaluation one step further, it should be possible to evaluate existing paradigms of AI. Again, an inductive approach based on AIQ tests and formal analyses of systems developed according the paradigms could be taken. Also, a deductive approach based on formal descriptions of AI paradigms and Universal Intelligence definition could be entertained. Let us now briefly consider requirements and possible results of such an endeavor.

1. Induction from AIQ test results and formal analyses:
    - As the approach is basically the same as in the previous section, only applied to a broader set of systems and paradigms, mentioned requirements and considerations also apply.
    - However, the issue of comparability of results is even more significant. Due to the way the Universal Intelligence definition and the derived AIQ test were constructed, the comparability issue should be taken care of. That is, as long as it is possible to test and analyze the systems from the point of view of their interaction with different environments, their different structure will not matter in any other way than in which it contributes to their performance.
    - The results can be used to compare existing paradigms of AI to each other. A hypothesis is held, that the hybrid paradigm is more suitable for reaching AGI. This is based on the fact that it synergically combines symbolic and connectionist paradigms as well as some philosophical considerations regarding the nature of the mind. That itself, obviously, cannot validate the hypothesis. An evaluation on this level, however, should make the validation possible.
    - The considered level of evaluation is high enough to contribute to the correctness of the very notion of functionalism. As the realization of the function according to functionalism does not matter, than if it is correct, the results should be generally the same for all the paradigms with some caveats. There is the recurrent issue of determining if the tested system represents well its paradigm, or if it is just poorly implemented, and if this could skew the results unfavorably for a certain paradigm. Also, if the results were not reasonably comparable, it would not disprove functionalism, as it may be that just some paradigm is poor.
    - The results could also contribute to the question of attainability of AGI, on condition that the concept is more precisely specified with respect to AIQ or

the Universal Intelligence. A possible way may be to use fuzzy intervals over AIQ values.

2. Deduction from formal description of AI paradigms:
   - Considerations for deduction mentioned by the previous section also apply.
   - Also, the comparability of results ensured mainly by the Universal Intelligence definition is more significant at this level.
   - Considered deductive approach can contribute as well to validation of the hypothesis of suitability of the hybrid paradigm for AGI.
   - The deduction can as well be instrumental in supporting the functionalism. Compared to the inductive approach, there would be a benefit in not needing to deal with whether a system is implemented well or not.
   - Similarly, the attainability of AGI could be somewhat enlightened.
   - Having both the results of the inductive approach as well as the deductive approach, their comparison can further clarify and support the finding in the above mentioned areas.

5 Conclusion and Future Work

The paper summarized philosophical and cognitive presumptions of intelligence in order to better grasp what artificial intelligence should strive for. The strong – weak and general – specific AI distinctions were described to illustrate the opposing interpretations of the main goal of the AI. Special attention was given to recent achievements in defining intelligence. The Universal Intelligence definition of Legg and Hutter [2007] was presented, as well as its approximation the AIQ test by Legg and Veness [2013].

Further, the paper described existing paradigms of AI. Beginning with traditional symbolism and biologically inspired connectionism, the paper especially noted the hybrid approach, since it focuses on finding a synergic combination of the two paradigms. Approaches of situated cognition which stress out the role of body and perception – action interaction of an agent with its environment were also noted.

Finally, the paper discussed possibilities of evaluating AI from the AGI point of view on different levels of abstraction. Basically the Universal Intelligence definition can be used for formal analysis of presumptions of intelligence, while the AIQ test can be used for practical testing. A specific system can be evaluated, or the focus can be more general on a specific paradigm or even on all paradigms. Using the AIQ test calls for an inductive approach with induction-specific limitations, while using the Universal Intelligence definition requires a deductive formal analysis based on a formal specification of a system or a paradigm. While the results of the AIQ test can be used to compare systems and paradigms, and also to compare incremental versions of them, the results cannot easily identify ways in which an extension of a system should be attempted. If an evaluation of all paradigms is undertaken, it can provide some clues for the validity of functionalism, however it cannot disprove it. Also, this level of evaluation can validate the hypothesis that hybrid paradigm is more suitable for AGI than other approaches. All in all it seems that focusing on evaluation of AI systems and paradigms can be worth of further research.

Mentioning that, there is future work to be done in several areas, ranging from practical to theoretical. Practical issues include identifying ways of choosing AI systems suitable for AIQ testing and specifying a set of criteria for the choice. Also, technical means of interfacing the system with the test should be found. On the theoretical side, the research should focus on devising interaction-based formal descriptions of systems so that they could be analyzed using the Universal Intelligence definition. Attention should be also given to finding the ways of transformation existing structural formalisms into interaction-based. Generalizations of such formalisms should be looked for to facilitate the analysis on the level of AI paradigms.

References

[Anderson *et al.*, 2004] John R. Anderson, Daniel Bothell, Michael D. Byrne, Scott Douglass, Christian Lebiere, and Yulin Qin. An integrated theory of the mind. *Psychological Review*, 111(4):1036–1060, Oct 2004.

[Besold *et al.*, 2015] Tarek Besold, José Hernández-Orallo, and Ute Schmid. Can machine intelligence be measured in the same way as human intelligence? *KI-Künstliche Intelligenz*, pages 1–7, 2015.

[Bringsjord and Schimanski, 2003] Selmer Bringsjord and Bettina Schimanski. What is artificial intelligence? psychometric ai as an answer. In *Proceedings of the 18th international joint conference on artificial intelligence (IJCAI'03)*, pages 887–893. Citeseer, 2003.

[Brooks, 1991] Rodney A Brooks. Intelligence without representation. *Artificial intelligence*, 47(1):139–159, 1991.

[Churchland and Churchland, 1990] Paul M. Churchland and Patricia Smith Churchland. Could a machine think? classical ai is unlikely to yield conscious machines; systems that mimic the brain might. *Scientific American*, 262(1):32–37, 1990.

[de Mey, 1992] Mark de Mey. *The cognitive paradigm*. University of Chicago Press, Chicago and London, 1992.

[Dennett, 1993] Daniel Clement Dennett. *Consciousness explained*. Penguin Books, 1 edition, 1993.

[Descartes, 1637] René Descartes. *A Discourse on Method*. Oxford University Press, Oxford, 1992 (1637).

[Goertzel, 2014] Ben Goertzel. Artificial general intelligence: Concept, state of the art, and future prospects. *Journal of Artificial General Intelligence*, 5(1):1–48, 2014.

[Gray, 2014] Norman Gray. Rdf, the semantic web, jordan, jordan and jordan. 2014.

[Harnad, 1990] Stevan Harnad. The symbol grounding problem. *Physica D*, pages 335–346, June 1990.

[Harnad, 1991] Stevan Harnad. Other bodies, other minds: A machine incarnation of an old philosophical problem. *Minds and Machines*, 1(1):43–54, 1991.

[Hebb, 1949] Donald Olding Hebb. *The organization of behavior*. Wiley, New York, 1949.

[Legg and Hutter, 2007] Shane Legg and Marcus Hutter. Universal intelligence: A definition of machine intelligence. *Minds and Machines*, 17(4):391–444, Dec 2007.

[Legg and Veness, 2013] Shane Legg and Joel Veness. An approximation of the universal intelligence measure. In *Algorithmic Probability and Friends. Bayesian Prediction and Artificial Intelligence*, pages 236–249. Springer, 2013.

[Maturana and Varela, 1979] Humberto Maturana and Francisco Varela. *Autopoiesis and cognition*. Boston studies in the philosophy of science ; v. 42. D. Reidel Pub. Co., 1979.

[McClelland *et al.*, 1986] James L McClelland, David E Rumelhart, PDP Research Group, et al. Parallel distributed processing. *Explorations in the microstructure of cognition*, 2, 1986.

[McCulloch and Pitts, 1943] Warren Sturgis McCulloch and WA Pitts. A logical calculus of the ideas immanent in neural nets. *Bulletin of Mathematical Biophysics*, 5:115, 1943.

[Merleau-Ponty, 1942] Maurice Merleau-Ponty. *The Structure of Behaviour*. Beacon Press, Boston, 1963 (1942).

[Merleau-Ponty, 1945] Maurice Merleau-Ponty. *Phenomenology of Perception*. Humanities Press, New York, 1962 (1945).

[Newell and Simon, 1976] Allen Newell and Herbert A Simon. Computer science as empirical inquiry: Symbols and search. *Communications of the ACM*, 19(3):113–126, 1976.

[Rapaport, 1995] William J. Rapaport. Understanding understanding: Syntactic semantics and computational cognition. *Philosophical perspectives*, 9:49–88, 1995.

[Rosenblatt, 1958] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[Schweizer, 2012] Paul Schweizer. The externalist foundations of a truly total turing test. *Minds and Machines*, 22(3):191–212, Aug 2012.

[Searle, 1980] John Rogers Searle. Minds, brains, and programs. *Behavioral and Brain Sciences*, 3(3):417–457, 1980.

[Sun *et al.*, 2005] Ron Sun, Paul Slusarz, and Chris Terry. The interaction of the explicit and the implicit in skill learning: A dual-process approach. *Psychological Review*, 112(1):159–192, Jan 2005.

[Sun, 2001] Ron Sun. *Artificial intelligence: Connectionist and symbolic approaches*, pages 783–789. Pergamon/Elsevier, Oxford, 2001.

[Sun, 2007] Ron Sun. The importance of cognitive architectures: An analysis based on clarion. *Journal of Experimental & Theoretical Artificial Intelligence*, 19(2):159–193, 2007.

[Thomsen, 2013] Knud Thomsen. The cerebellum in the ouroboros model, the "interpolator hypothesis". In Shunji Shimizu and Terry Bossomaier, editors, *COGNITIVE 2013, The Fifth International Conference on Advanced Cognitive Technologies and Applications*, pages 37–41, Valencia, Spain, 2013. IARIA.

[Turing, 1936] Alan Mathison Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42):230–265, 1936.

[Turing, 1950] Alan Mathison Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.

[Vadinský, 2013] Ondřej Vadinský. Towards an artificially intelligent system: Philosophical and cognitive presumptions of hybrid systems. In Shunji Shimizu and Terry Bossomaier, editors, *COGNITIVE 2013, The Fifth International Conference on Advanced Cognitive Technologies and Applications*, pages 97–101, Valencia, Spain, 2013. IARIA.

[Vadinský, 2014] Ondřej Vadinský. *Na cestě k uměle inteligentním systémům: Intencionalita a percepčně-akční spirála v hybridních systémech*, pages 217–222. Slezská univerzita, Opava, 1 edition, 2014.

[Walter, 2010] Sven Walter. Locked-in syndrome, bci, and a confusion about embodied, embedded, extended, and enacted cognition. *Neuroethics*, (3):61–72, 2010.