Reversibility reconsidered: finite-state factors for efficient probabilistic sampling in parsing and generation

Marc Dymetman[†], **Sriram Venkatapathy**[‡], **Chunyang Xiao**[†]

[†]Xerox Research Centre Europe, Grenoble, France
[‡]Amazon, Machine Learning Team, Bangalore, India[*]
[†]{*first.last*}@xrce.xerox.com, [‡]vesriram@amazon.com

Abstract

We restate the classical logical notion of generation/parsing reversibility in terms of feasible probabilistic sampling, and argue for an implementation based on finite-state factors. We propose a modular decomposition that reconciles generation accuracy with parsing robustness and allows the introduction of dynamic contextual factors. *(Opinion Piece)*

1 Introduction

The objective of Natural Language Understanding (NLU) is to map linguistic utterances to semantic representations, that of Natural Language Generation (NLG) to map semantic representations to linguistic utterances. In most of NLP practice, these two objectives are handled by different processes, and computational linguists rarely operate at the intersection of the two subdomains.

For a few years around the early nineties, based both on cognitive, linguistic, and engineering considerations, there was a surge of interest in so called *reversible grammar* approaches to NLP, where one and the same grammatical specification could serve both for parsing utterance x into logical form z, but also for generating x from z (Strzalkowski, 1994).

We start by a brief review of this historical non-probabilistic notion of reversibility and point out certain of its weaknesses, in particular regarding robustness; we then give in section 3 a new probabilistic definition of reversibility; then, in section 4 we argue for a reversibility model based on modular weighted finite-state transducers. We end with a discussion of recent related work.

[*] Work done while at XRCE.

2 Classical reversibility

The most direct approaches to NLU attempt to design procedures for semantic parsing that, given an input utterance x, produce a semantic representation z, by following a number of intermediate steps where the surface form is gradually transformed into semantic structure. Such “procedural” approaches to semantic parsing are typically very hard or impossible to invert: starting from a semantic representation z, there is no simple process that is able to find an x which, when given to the parser, would produce z. Formally, a Boolean relation $r(x, z)$ can be such that the question $?\exists z\ r(x, z)$ is decidable for all x's, while the reciprocal question $?\exists x\ r(x, z)$ is undecidable for some z's (Dymetman, 1991).[1] One of the motivations for the emerging paradigm of unification grammars at the end of the eighties was the clean separation they promised between specifying well-formed linguistic structures, both on the syntactic and semantic levels, through a formal description of the relation $r(x, z)$, and producing efficient implementations of the specification; in particular, there was much hope that such formalisms would be conductive to effective reversibility (by contrast to variable assignment, variable unification is inherently symmetrical), that is, to feasible (and if possible efficient) implementations of the parsing problem $r(x, ?)$ and of the generation problem $r(?, z)$.

To some extent, this hope was validated through a number of works at the time, mostly involving machine translation applications, and constraining in more or less explicit ways the specification of r (van Noord, 1990). However, for the non-statistical approaches to parsing then strongly dominant, *robustness* was an issue: a parser had to

[1] Some intuition into the issue may be gained by considering typical techniques of public key cryptography, which rely on the difficulty of inverting some simple arithmetic computations.

either accept or reject a given input x, with no intermediary options, and in order to be able to parse actual utterances, with all their empirical diversity, parsers had to be rather tolerant. In the procedural view of parsing, such robustness issues could often be mitigated through engineering tricks such as ordering the rules from strict to lax, where grammatical constructions were given preference over less conventional ones; however, when trying to move to reversible grammars, these tricks could not be reproduced: if the grammar was able to parse an x into z, then, by design, it was also able to generate x from z, and there was no obvious way, in these non-probabilistic approaches, to distinguish between producing a linguistically correct x or producing a deviant or incorrect one.

3 Probabilistic reversibility

In the classical non-probabilistic case, a (relative) consensus existed around the fact that a reversible grammar should be, as we indicated above, a formal specification of the relation $r(x, z)$ such that the problems $r(x, ?)$ and $r(?, z)$ were effectively solvable.

Transposing this to the probabilistic world, we propose the following semi-formal Definition:

A probabilistic reversible grammar is a formal specification of a joint probability distribution $p(x, z)$ over logical forms z and utterance strings x such that the conditional distributions $p(z|x) \overset{\text{def}}{=} \frac{p(x,z)}{\sum_{z'} p(x,z')}$ (parsing) and $p(x|z) \overset{\text{def}}{=} \frac{p(x,z)}{\sum_{x'} p(x',z)}$ (generation) can be efficiently sampled from.[2]

Why such focus on *sampling*? We could have chosen other definitions of parsing (and similarly for generation), for instance the ability to return the *most probable* z given x, i.e. to return $\text{argmax}_z\, p(z|x)$; however sampling is the most direct way of providing a concrete view of the underlying probabilistic distribution, and has many applications to learning, so we think the definition above is reasonable (see also footnote[4]).

[2]We note the "semi-formal" aspect of this definition: contrarily to the classical case, which has a formal notion of effective computation, there is no universally accepted notion of effective sampling from a probability distribution. For many probability distributions, the only feasible sampling approaches are the *MCMC* techniques (Robert and Casella, 2004), which typically do not come with convergence guarantees; in some situations, *exact sampling* techniques are applicable, which come with much better guarantees. We will see that the approach proposed in section 4 allows such exact sampling to take place.

4 Finite-state models for reversibility

Finite-state transducers have properties which make them uniquely suited to implementing reversible linguistic specifications in the above sense. Consider a simple weighted string-to-string transducer $\tau(s, t)$, where s, t are strings, and where the underlying semiring is the "probabilistic semiring" over the nonnegative reals, addition and multiplication having their usual interpretations. Such a transducer *preserves regularity*, both in the forward (resp. reverse) directions, meaning that the image through τ of any weighted regular language over s (resp. over t) is again a weighted regular language over t (resp. over s). In particular the forward (resp. reverse) image of a fixed string s_0 (resp t_0) can be computed *in a compact form* as a weighted finite-state automaton (FSA) over t (resp. s), which we can denote by $\tau(s_0, \cdot)$ (resp. $\tau(\cdot, t_0)$). A weighted FSA can be easily normalized into a probabilistic FSA[3] and, from this probabilitic FSA exact samplers for the "parser" $\tau(s_0, \cdot)$ and for the "generator" $\tau(\cdot, t_0))$ are directly obtained.[4]

In general, some of the properties that make weighted FSAs and FSTs — over strings or trees — specially relevant for probabilistic models of language are the following: (i) they allow compact representations of complex probability distributions over linguistic objects (automata) or pairs of linguistic objects (transducers), (ii) they permit efficient exact sampling (and efficient optimization over derivations (but not always over strings)), (iii) they support modularity: intersection of automata, composition of transducers, projections of an automaton through a transducer.[5]

Conceptual architecture Armed with these general considerations, let us now propose a conceptual architecture based on a small number of

[3]That is, into a weighted FSA such the weights of the transitions from each state sum to 1.

[4] While *sampling* strings from a weighted finite-state automaton is simple, finding the most probable *string* (not *path*) in a probabilistic FSA is an NP-hard problem (Casacuberta and de la Higuera, 2000), and one has to resort to the so-called Viterbi approximation (assuming that the most probable path projects into the most probable string). Contrary to popular belief, sampling can sometimes be simpler than optimization.

[5]Outside of the realm of finite-state machines, this modularity is typically impossible to obtain. Thus, in general, the availability of a sampler for a distribution $p(x)$ (resp. a distribution $q(x)$) does not imply that we can efficiently sample from the product (i.e. intersection) $p(x).q(x)$, but we can in case p and q are both represented by weighted FSAs.

finite-state modules, which attempts to satisfy the definition given above for probabilistic reversibility, to address the problem of robustness that we described earlier, and can also support contextual preferences. We illustrate the approach with some simple examples of human-machine dialogues (between a customer and a virtual agent), a domain for which reversibility has high relevance, due to effects such as self-monitoring (Neumann, 1998; Levelt, 1983), interleaving of understanding and generation (Otsuka and Purver, 2003), and lexical entrainment (Brennan, 1996).

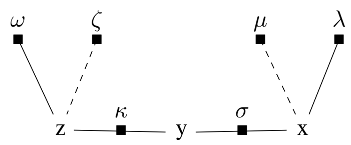

Figure 1: Reversible specification through finite-state factors.

The conceptual architecture is shown in Figure 1. Formally, the figure represents a probabilistic graphical model in so-called factor form, where the factors are $\omega, \kappa, \sigma, \lambda$ (we have also indicated for future reference the "contextual" factors ζ, μ, that we ignore for now). The factors take as arguments three types of objects: z is a logical form, that is, a structured object which can be naturally represented as a tree, x is a surface string, and y is a latent "underlying" string that corresponds to one of a small collection of "canonical" texts for realizing the logical form z (more about that later).

Each factor is realized through a weighted finite-state machine (acceptor or transducer) over strings or trees (Mohri, 2009; Fülöp and Vogler, 2009; Maletti, 2010; Graehl et al., 2008).

The λ factor is a string automaton that represents a standard ngram language model (typically specific to domain), in other words a probability distribution over utterances x. *Symmetrically*, the regular tree automaton ω represents a distribution over logical forms z, which can be seen as playing a similar role to the language model, but at the semantic level, namely telling us what are the possible/likely logical forms in a certain domain.[6]

The "canonical factor" κ is a weighted tree-to-string transducer (Graehl et al., 2008), which implements a relation between logical forms z and a small number of latent "canonical" texts y realizing these logical forms. For example, κ may associate the logical form (dialog act) $z = \mathtt{wad}(\mathtt{batLife}, \mathtt{iphone6})$ — with `wad` an abbreviation for "what is the value of this attribute on this device?", and `batLife` an abbreviation for "battery life" —, with such a canonical text (among a few others) as: *What is the battery life of the Iphone 6?*.

The "similarity factor" σ is a weighted string-to-string finite state transducer which gives scores to x, y according to a notion of similarity. It has the role of "bridging" the gap between the actual utterances x and the latent canonical utterances y. The intention behind the similarity factor is to "decouple" the task of modeling some possible realizations of a given logical form from the task of recognizing that a given more or less well-formed input is a variant of such a realization. This factor relates the two strings y and x, where y is a possible canonical utterance in the limited repertory produced by κ, and x is an actual utterance, in particular any utterance that could be produced by a human speaker. So for instance suppose that the user's utterance is $x =$ *What about battery duration on this Iphone 6?*, we would like this x to have a significant similarity with the canonical utterance $y =$ *What is the battery life of the Iphone 6?* but a negligible similarity with another canonical utterance such as $y' =$ *What is the screen size of the Galaxy Trend?*.

Overall, the canonical factor $\kappa(z, y)$ concentrates more on a core "generation model", namely on producing some well-formed output y from a logical form z, while the similarity factor $\sigma(y, x)$ allows relating an actual user input x to a possible output y of the κ model. The main import of σ is then to allow to use the core generation *model defined by κ to be exploited for* robust semantic parsing.

Different instantiations of this scheme can be employed. In some preliminary experiments that we have performed,[7] σ is a simple edit-distance transducer (Mohri, 2003) which penalizes differently the discrepancies between x and y: strongly for some salient content words or named entities of the domain, weakly for less relevant content words and for non-content words, with limited use of local paraphrases (which can also be implemented

[6]In particular, the ω factor makes explicit the notion of a well-formed semantic representation, a notion often left implicit in semantic parsing.

[7]In these experiments, we used string-based approximations of the logical forms, and only employed string-based transducers from the OpenFST library.

through σ). This strategy seems to work reasonably well when the semantical repertory of the domain is restricted, because a large number of possible variants for x are "attracted" to the same underlying semantics. In domains where small nuances of expression may result in distinct semantics, the division of work between κ and σ may be different.

Parsing and Generation To understand the reversibility properties of the model of Figure 1, let us first simplify the description by assuming that z, instead of being a tree, is actually a string. Then both ω and λ are string automata, and both κ and σ string-to-string transducers. Such a specification satisfies our definition of probabilistic reversibility, exploiting well-known compositionality properties of weighted finite-state machines over strings (Mohri, 2009). For parsing, we start from a fixed x_0, and can project it through σ into a weighted FSA over y; in turn we can project this automaton onto an FSA over z, and finally intersect this automaton with ω, obtaining a final weighted "x_0-parser" automaton over z, representing a probability distribution from which we can draw exact samples as explained above.[8] Generation works in exactly the reverse way, starting from a z_0 and eventually building a "z_0-generator" automaton over x.

In the actual proposal, z is a tree, meaning that ω is a tree automaton, and κ a tree-to-string transducer. While finite-state tree automata correspond to a single concept, and share all the nice properties of string automata (Comon et al., 2007), the situation with tree-to-tree or tree-to-string transducers is more complicated (Maletti, 2010; Graehl et al., 2008): several variants exist, only some of which support the operations that our conceptual model requires (composition with the string transducer σ and intersection with the tree automaton ω). In particular, the "linear non-deleting top-down tree transducers" defined in (Maletti, 2010)[9] have the requisite properties.

Contextual factors We now briefly come back to the factors ζ (tree automaton) and μ (string automaton) of Figure 1, which highlight the usefulness of our modular finite-state architecture. These factors play similar roles to ω and λ, but they evolve dynamically with the context. In dialogue applications, utterances can often only be interpreted by reference to the current dialogue state (e.g. "ten hours" in the context of a question about battery life), and the ζ factor can be used as a compact representation of the current expectations of the dialogue manager about the next logical form, to be combined with the actual customer's utterance. Symmetrically, the μ factor can be used to represent such phenomena as lexical entrainment (Brennan, 1996), where the agent's utterance is oriented towards using similar wordings to the customer's.

[8]We could also have precompiled a generic parser for all x's by first marginalizing the latent variable y through a composition of the transducers κ and σ, and then intersecting the resulting transducer with the automaton ω.

[9]The paper only defines tree-to-tree transducers, but tree-to-string variants can be derived easily.

5 Related work

The unique formal properties of finite-state machines, which favor modular decompositions of complex tasks, have long been exploited in Computational Linguistics. Tree transducers in particular have gained popularity in Statistical Machine Translation, starting with (Yamada and Knight, 2001), as described in the surveys (Maletti, 2010; Razmara, 2011).

The *reversibility* properties of finite-state transducers have been exploited to a more limited extent, starting with applications of non-weighted string-to-string transducers to morphological analysis and generation (Beesley, 1996).

Concerning the application of weighted finite-state tree machines to NLU/NLG reversibility, our proposal is strongly related on the one hand to the approach of (Jones et al., 2012), who explicitely proposes tree-to-string transducers as a tool for modelling semantic parsing and for training on semantically annotated data, and on the other hand to (Wong, 2007; Wong and Mooney, 2007), who focus more directly on the problem of inverting a semantic parser into a generator. Wong et al. do not explicitely use tree-based transducers, but rather a formalism inspired by SCFGs (synchronous context-free grammars), which essentially corresponds to a form of tree-to-string transducer. In relation to reversibility considerations, presentations in terms of synchronous formalisms have the interest that they are intrinsically symmetrical. Such formalisms have tight relations to tree-transducers (Shieber, 2004); one recently proposed generalization, "Interpreted Regular Tree Grammars" (Koller and Kuhlmann, 2011), allows multiple (possibly more than two) synchronized views of an underlying abstract derivation tree, and has the advantage of permitting a uniform treatment of strings and trees.

One important aspect in which our proposal differs from these previous approaches is in proposing to decouple the "core" task of mapping logical forms to well-formed latent canonical realizations from the task of relating these realizations to actual utterances, through an additional "similarity" transducer acting as a bridge.

This idea of a bridge is however close to another line of work in semantic parsing, not transducer based, namely (Berant and Liang, 2014; Wang et al., 2015). There, a simple generic grammar is used to generate canonical realizations from a repertory of possible logical forms (expressed in a variant of lambda calculus). Given an input to parse, simple heuristics are used to select a finite list of potential logical forms which are then ranked according to the (paraphrase-based) similarity of their associated canonical realization with the input. Thus in this approach, a form of generation plays an important role, not for its own sake, but as a tool for semantic parsing.

6 Conclusion

Because of their unique compositional properties, finite-state modules are a natural choice for implementing our definition of reversibility as efficient bidirectional sampling from a common specification. In this piece we have argued in favor of an architecture realizing this definition and displaying robustness and contextuality.

Acknowledgments We thank the anonymous reviewers for their detailed comments and for pointing us to some relevant literature that we had overlooked.

References

Kenneth Beesley. 1996. Arabic Finite-State Morphological Analysis and Generation. In *Coling*, pages 89–94.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1415–1425, Baltimore, Maryland, June. Association for Computational Linguistics.

S.E. Brennan. 1996. Lexical entrainment in spontaneous dialog. In *Proceedings of International Symposium on Spoken Dialogue (ISSD-96)*.

Francisco Casacuberta and Colin de la Higuera. 2000. Computational complexity of problems on probabilistic grammars and transducers. In *ICGI*, pages 15–24.

H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. 2007. Tree automata techniques and applications. Available on: `http://www.grappa.univ-lille3.fr/tata`. release October, 12th 2007.

Marc Dymetman. 1991. Inherently reversible grammars, logic programming and computability. In *In Proceedings of the ACL Workshop: Reversible Grammar in Natural Language Processing*.

Zoltán Fülöp and Heiko Vogler. 2009. Weighted tree automata and tree transducers. In Manfred Droste, Werner Kuich, and Heiko Vogler, editors, *Handbook of Weighted Automata*, Monographs in Theoretical Computer Science. An EATCS Series, pages 313–403. Springer Berlin Heidelberg.

Jonathan Graehl, Kevin Knight, and Jonathan May. 2008. Training tree transducers. *Comput. Linguist.*, 34(3):391–427, September.

Bevan Keeley Jones, Mark Johnson, and Sharon Goldwater. 2012. Semantic parsing with bayesian tree transducers. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 488–496, Stroudsburg, PA, USA. Association for Computational Linguistics.

Alexander Koller and Marco Kuhlmann. 2011. A generalized view on parsing and translation. In *Proceedings of the 12th International Conference on Parsing Technologies*, IWPT '11, pages 2–13, Stroudsburg, PA, USA. Association for Computational Linguistics.

W.J.M Levelt. 1983. Monitoring and self-repair in speech. *Cognition*, 14(1):41–104.

Andreas Maletti. 2010. Survey: Tree transducers in machine translation. Technical report, Universitat Rovira i Virgili.

Mehryar Mohri. 2003. Edit-Distance of Weighted Automata: General Definitions and Algorithms. *International Journal of Foundations of Computer Science*, 14:957–982.

Mehryar Mohri. 2009. Weighted automata algorithms. In Manfred Droste, Werner Kuich, and Heiko Vogler, editors, *Handbook of Weighted Automata*, pages 213–254. Springer.

Günter Neumann. 1998. Interleaving natural language parsing and generation through uniform processing. *Artificial Intelligence*, 99(1):121–163.

M. Otsuka and M. Purver. 2003. Incremental generation by incremental parsing. In *Proceedings 6th CLUK Colloquium.*

Majid Razmara. 2011. Applications of tree transducers in statistical machine translation. Technical report, Simon Fraser University.

Christian P. Robert and George Casella. 2004. *Monte Carlo Statistical Methods (Springer Texts in Statistics).* Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Stuart M. Shieber. 2004. Synchronous grammars as tree transducers. In *Proceedings of the Seventh International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+ 7)*, Vancouver, Canada, 20–22 May.

Tomek Strzalkowski, editor. 1994. *Reversible Grammar in Natural Language Processing*. Springer.

Gertjan van Noord. 1990. Reversible unification based machine translation. In *Proceedings of the 13th Conference on Computational Linguistics - Volume 2*, COLING '90, pages 299–304, Stroudsburg, PA, USA. Association for Computational Linguistics.

Y. Wang, J. Berant, and P. Liang. 2015. Building a semantic parser overnight. In *Association for Computational Linguistics (ACL).*

Yuk Wah Wong and Raymond J Mooney. 2007. Generation by inverting a semantic parser that uses statistical machine translation. In *HLT-NAACL*, pages 172–179.

Yuk Wah Wong. 2007. *Learning for Semantic Parsing and Natural Language Generation Using Statistical Machine Translation Techniques*. Ph.D. thesis, Department of Computer Sciences, University of Texas at Austin, Austin, TX, August. Also appears as Technical Report AI07-343, Artificial Intelligence Lab, University of Texas at Austin, August 2007.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, ACL '01, pages 523–530, Stroudsburg, PA, USA. Association for Computational Linguistics.