Pairing Model-Theoretic Syntax and Semantic Network for Writing Assistance

Jean-Philippe Prost and Mathieu Lafourcade

LIRMM – 161, rue Ada – 34095 Montpellier Cedex 5 – France
{Prost,Lafourcade}@lirmm.fr

Abstract. In this paper we investigate the possibility of a syntax–semantics inferface between a framework for Model-Theoretic Syntax on one hand and a semantic network on the other hand. We focus on exploring the ability of such a pairing to solve a collection of grammar checking problems, with an emphasis on cases of missing words. We discuss a solution where constraint violations are interpreted as grammar errors and yield the re-generation of new candidate parses (partially unrealised) through tree operations. Follows a surface realisation phase, where missing words are filled through semantic network exploration.

1 Introduction

Model-Theoretic Syntax (MTS) refers to a family of frameworks for formal syntax, which is grounded in the Model Theory — unlike Generative-Enumerative Syntax (GES), such as the Chomskyan syntax, which originates from Proof Theory. In very general terms, an MTS framework considers that a (given) syntax structure is a model of the grammar $\mathcal{G}$, seen as a set of independent logical statements, if and only if it meets every statement in $\mathcal{G}$. For what we are interested in in this paper, it is important to note that: (i) the two problems of model generation and model recognition are kept separate, and (ii) every grammar statement may be evaluated independently from the rest of the grammar.

Many of the differences between the two families with respect to linguistic knowledge representation have been discussed by Pullum and Scholz (21). The ability of MTS, unlike GES, to represent linguistic information about quasi-expressions[1], that is, utterances which present grammar irregularities, is the main point of interest here. As we are going to show it, that linguistic knowledge which we have about the syntax of quasi-expressions directly serves the purpose of error detection and correction.

Meanwhile, a drawback of MTS for phrase structure grammar is a lack of syntax–semantics interface, which, beyond compositionality, would take advantage of the fine-grained information made available next to the phrase structure. To the best of our knowledge the work from Dahl and Gu (9) is the only exception. It extends Property Grammar (Blache 4) with semantic predicates, which further constrain the specification of categories as any other assertion. Those

[1] Sometimes also referred to as *non-canonical input* in the literature.

semantic predicates essentially introduce different kinds of semantic relationships among lexical items. Those are mostly domain-specific, derived from a biomedical ontology. A semantic form is built through the parsing process by compositionality. Meanwhile, there seems to be no attempt to take advantage of the linguistic knowledge embodied in every property satisfied or violated by the parse tree.

In this paper, we address the same problem of the syntax–semantics interface for Property Grammar (PG) but from a different angle; we investigate the possibility to pair an MTS framework with a semantic network. We explore, especially, how such a pairing can serve the purpose of grammar error correction, and focus on cases of missing words. We show how, beyond compositionality, the constraints can interact with the phrase structure to build a more detailed semantic representation than with the phrase structure alone. Starting from an approximated parse for a quasi-expression missing a word, the process operates specific tree transformations according to the detected error, in order to generate a new set of candidate corrected trees. The characterisation of those candidate models let us build the messages with which the semantic network is then queried, in search for the missing word.

In section 2 we present briefly the theoretical background we are working with; in section 3 we detail how we adapt that theoretical framework to the semantic roles required by the semantic network; then in section 4 we present how an approximated parse is turned into a candidate corrected parse by regeneration; finally, section 5 describes how to explore a semantic network in order to fill missing words.

2 Property Grammar

The framework we are using for knowledge representation is Property Grammar (Blache 4)[2] (PG), for which a model-theoretical semantics was axiomatised by Duchier et al. (11). Intuitively, a PG grammar decomposes what would be rewrite rules of a generative grammar into atomic syntactic properties — a *property* being represented as a boolean constraint. Take, for instance, the rewrite rule $\texttt{NP} \rightarrow \texttt{D N}$. That rule implicitly informs[3] on different properties (for French): (1) NP has a D child; (2) the D child is unique; (3) NP has an N child; (4) the N child is unique; (5) the D child precedes the N child; (6) the N child requires the D child. PG defines a set of axioms, each axiom corresponding to a constraint type. The properties above are then specified in the grammar as the following constraints: (1) $\texttt{NP} : \triangle\texttt{D}$; (2) $\texttt{NP} : \texttt{D!}$; (3) $\texttt{NP} : \triangle\texttt{N}$; (4) $\texttt{NP} : \texttt{N!}$; (5) $\texttt{NP} : \texttt{D} \prec \texttt{N}$; (6) $\texttt{NP} : \texttt{N} \Rightarrow \texttt{D}$. A PG grammar is traditionally presented as a collection of Categories (or Constructions), each of them being specified by a set of constraints. Table 1 shows an example of categories.

[2] Property Grammars closely follows from the 5P Paradigm introduced by Bès and Blache (3).

[3] The rule is assumed to be the only one for NP.

These constraints can be independently verified, hence independently satisfied or violated. The parsing problem is, thus, a Constraint Satisfaction Problem (CSP), where the grammar is the constraint system to be satisfied. In the Model-Theoretic (MT) axiomatisation the class of models we are working with is made up of trees labelled with categories, whose surface realisations are the sentences σ of the language. A syntax tree of realisation the expression (i.e well-formed sentence) σ is a *strong* model for the PG grammar $\mathcal{G}$ iff it satisfies every pertinent constraint[4] in $\mathcal{G}$.

The loose semantics also allows for constraints to be relaxed. Informally, a syntax tree of realisation the quasi-expression (i.e. ill-sentence) σ is a *loose* model for $\mathcal{G}$ iff it maximises the proportion of satisfied constraints in $\mathcal{G}$ with respect to the total number of pertinent ones (i.e. evaluated) for a given category. Such a loose model is called an *approximated parse*.

The set of all the satisfied constraints and all the violated ones for a model $\mathcal{M}$ is called the model's *characterisation*. It provides fine-grained information about every node in the model, which complements usefully the sole phrase structure. The violated constraints, especially, naturally point out grammar errors.

On the downside, although the formal model is quite elegant in practice the size of the search space makes the parsing problem blow up exponentially. As emphasized by Duchier et al. (10), who implemented a parser as a genuine CSP based on the MT axiomatisation, the practical issue is not so much finding the best model (for which existing constraint programming techniques are quite efficient) as proving its optimality. The reason for that comes from the need to explore the entire class of models in order to address the decision problem. Yet, such an implementation does actually keep separate the generation of models from their checking, which opens the door to different, and more efficient, perspectives.

One of them is to reduce the search space to the subset of models, which are statistically the most significant. It can easily be achieved by any stochastic robust parser, or even a combination of them. That subset can even be completed with models for which the statistical significance is unknown, but which were generated by symbolic parser according to linguistic judgements. The Sygfran parser (Chauché 7), for instance, is one of the latter. Provided such a small subset as search space the combinatorial explosion is avoided. Each model can, then, easily be completed with its characterisation. Incidently, it is interesting to notice that such an architecture makes it possible to overcome an important decision problem met by statistical robust parsing with respect to the grammaticality of sentence, and despite the fact that a parse tree is generated for it. That problem makes authors such as Wagner et al. (24) or Wong and Dras (25), among others, say that those robust parsers are *too* robust. Once the parses are characterised the decision about the grammaticality of a sentence is straightforward.

[4] As defined by Duchier et al. (11), a *pertinent constraint* is an instance of a property, which verifies the *pertinence predicate* $\mathcal{P}_\tau$. Intuitively, an instance of a property (or *constraint*) is pertinent at a node if the node's category and those of its children are in use in the property's definition; the constraint is otherwise trivially satisfied.

Another incentive is that it makes it a framework for comparing different parsers' outcomes over the same corpus. This is particularly interesting when it comes to quasi-expressions, for which there exists neither a linguistic theory nor an empirical consensus as to what an approximated parse should be. Further works should consider running such a comparison and addressing the question of parse quality, more especially on a corpus with a large number of ill-sentences.

3 Functional and Semantic Roles

The *Dependency* property in PG is the (only) property meant to model a semantic relationship between two constituents. According to the definition given in Blache (5), the dependeny property[5] $\texttt{c0} : \texttt{c1} \rightsquigarrow \texttt{c2}$ holds for the parent constituent of category `c0` between two children constituents of categories `c1` and `c2` if those two children constituents are *semantically compatible*. Intuitively, the role of that property in the grammar (still in (Blache 5)) is first to specify the existence of a predicative structure for the governor, and second to constrain the argument structure through the constituents feature structures.

That rather permissive definition is unclear as how the *semantic compatibility* between categories must be checked. The lack of conditional satisfaction makes it a property which can not be violated: either it holds true for two constituents, or it is non-pertinent, but it never fails. The rationale in Blache's proposal for such a semantics is to include in the characterisation of a sentence pieces of information regarding the existence of dependency relationships among constituents. VanRullen (23) takes a different perspective and defines for the Dependency property a semantics where the dependency relation between two categories is conditioned by feature agreement (e.g. in gender, number, person, ...) through feature unification.

In (Duchier et al. 10) the Dependency property is not axiomatised due to that lack of conditional satisfaction. Yet, the idea to combine phrase structure and dependency structure within the same information structure is elegant and quite convenient for interfacing syntax and semantics. Therefore we follow Blache's proposal on the Dependency property, which we modify slightly in order to be able to specify functional roles within a sentence and the way they propagate through the phrase structure. The property is conditioned either by the sole existence of the governor and the modifier nodes, as in the original proposal, or by feature values. Unlike in Blache, roles are feature values as opposed to features. Predicate subcategorisation schemes, especially, may serve to specify different arguments. The schemes for the verb *plaider* (*to plead*), given in Figure 1, come from the LexSchem resource (Messiant 19). Each argument category and role is then propagated in the phrase structure through Dependency properties. This way, it is possible to use the Dependency property to infer semantic roles from functional ones. In Table 1, for instance, the `VP` category specifies a Dependency between a `V` and an `NP`, where the `NP`'s semantic role is `PAT` (patient) whenever

[5] We use the notation introduced by Duchier et al. (11).

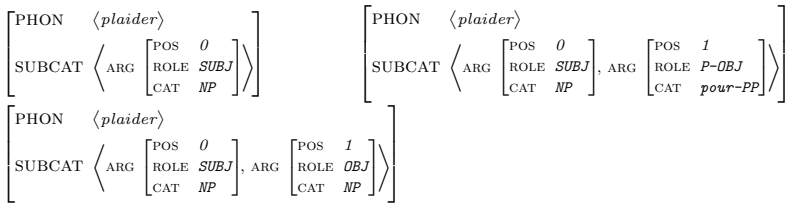

Fig. 1. Subcategorisation schemes for the verb *plaider* (*to plead*).

the `V` expects an object. Note that this should be refined whenever possible depending on the domain of values available for the semantic roles. A domain, for instance, which would make a distinction between a *patient* and a *recipient* role would benefit from a Dependency property where a `P-OBJ` (*for-object*) is turned into a recipient role.

4 Re-generation and Completion Message

Given a quasi-expression, an approximated parse tree for it, and its characterisation, is it possible to automatically infer candidate corrected syntax trees? To address that question we consider that every constraint failure in the characterisation can be interpreted positively either as a tree transformation operation or as an operation over feature structures. Those operations can be seen as grammar corrections.

Re-generation Take the example from Figure 2, where the constituent `NP2` violates the constraint `NP` : `N` ⇒ `D` (*within an NP constituent an N requires a D*). In order for that constraint to be satisfied it is sufficient to transform the `NP2`

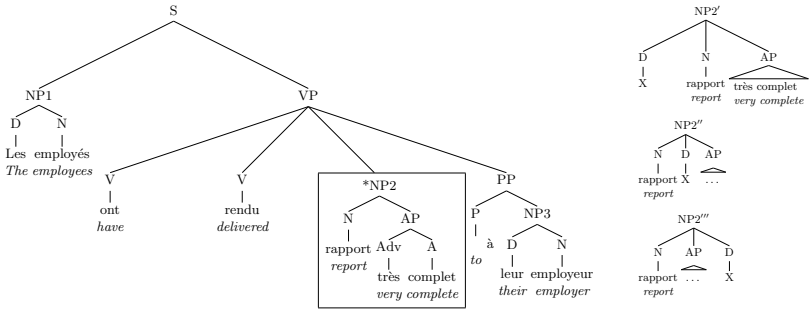

Fig. 2. Approximated parse for quasi-expression, and re-generated sub-trees

node by insertion of a daughter node labelled with the category `D`. This operation generates three sub-trees, illustrated in Figure 2. Three new candidate trees are then obtained by replacement of the `NP2` node by each of the three sub-trees.

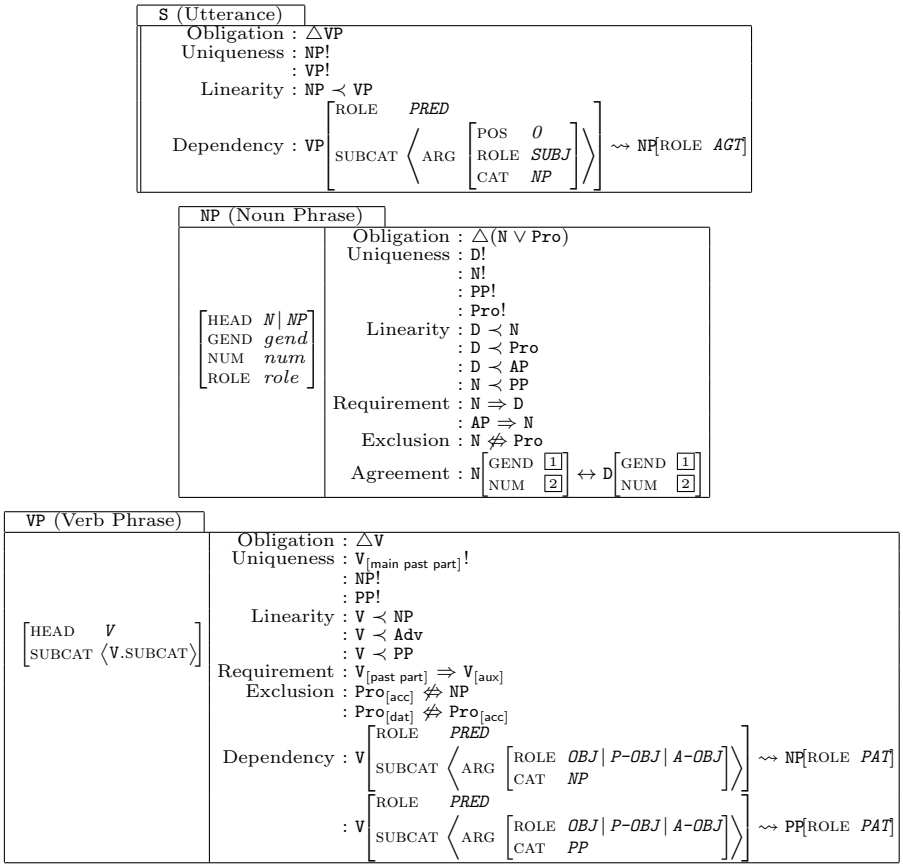

Table 1. Example property grammar for French

The multiple results are not a problem in that case, since they can easily be disambiguated through a new check of the grammar constraint system, that is, through the characterisation of each of the three new models. The sub-tree NP2′ is the only one meeting all the grammar constraints.

The sets of candidate sub-trees are generated using the following tree operations, where τ is a tree, and c, c_1, c_2 are node labels (i.e. categories):

- *Node insertion*, denoted by $\tau \downarrow c$
- *Node deletion*, denoted by $\tau \nmid c$
- *Node permutation*, denoted by $c_1 \overset{\tau}{\leftrightarrow} c_2$

After generalisation, every PG property corresponds to a transformation tree operation:

Property	*Violated instances*	*Tree operation*
Requirement	$\mathcal{I}_\tau[\![c_0 : c_1 \Rightarrow s_2]\!]$	$\tau \downarrow s_2$
Obligation	$\mathcal{I}_\tau[\![c_0 : \triangle c_1]\!]$	$\tau \downarrow c_1$
Linearity	$\mathcal{I}_\tau[\![c_0 : c_1 \prec c_2]\!]$	$c_1 \overset{\tau}{\leftrightarrow} c_2$
Uniqueness	$\mathcal{I}_\tau[\![c_0 : c_1!]\!]$	$\tau \nmid c_1$
Exclusion	$\mathcal{I}_\tau[\![c_0 : c_1 \not\Rightarrow c_2]\!]$	$\tau \nmid c_1 \cup \tau \nmid c_2$

Completion Message After re-generation and re-characterisation partial phrase structures are obtained, which contain underspecified nodes, including empty leaves (denoted by `X` in our figures). Those correspond to lexical items to be found through exploration of the semantic network. In the following, to illustrate the process we narrow down the scope of investigation and focus on cases of missing predicate. Figure 3 illustrates a candidate re-generated model for the quasi-expression **L'avocat le dossier de son client* (**The lawyer his client's file*). In order to complete the `NP2`'s surface realisation the exploration process of the

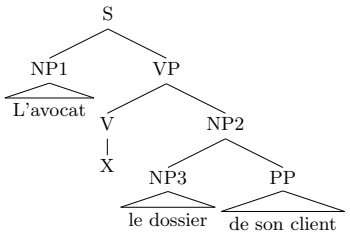

Fig. 3. Candidate re-generated model for quasi-expression

semantic network is fed with messages about the semantic relationships the missing predicate is involved in. Those messages take the form of triples $\langle a, \texttt{:R}, b\rangle$, where `:R` denotes an oriented semantic relation, and a and b its ordered elements. Examples of relations in use in the *rezoJDMFR* network (Lafourcade and Joubert 16) are Agent, Patient, Instrument, Succession, ..., though at this stage we limit ourselves to the Agent and Patient ones. The messages are built from gathering the relevant information in the model's characterisation.

Following up with the example sentence from Figure 3, we know from `VP` that `NP2` is in a Patient relationship with the predicate `V`, which gives us a value for R in message ♯1. `NP2` takes its head from `NP3` by propagation, which takes its own from the noun *dossier* (*file*). That gives us a value for a in message ♯1. The last message member is instantiated with the wildcard `X`, hence the message ♯1: $\langle \texttt{X}, \texttt{:PAT}, \textit{dossier}\rangle$. As for the members of message ♯2, they come

from the knowledge in S that NP1 is in an Agent relationship with VP, which by inheritance let us instantiate b with the value *avocat* (*lawyer*). That gives us, for message ♯2, $\langle$*avocat*, :AGT, X$\rangle$.

In the end we get the following list of messages:
$\{\langle$X, :PAT, *dossier*$\rangle, \langle$*avocat*, :AGT, X$\rangle\}$.

5 Propagation

For our experiments, we make use of a lexical network that has been constructed by means of a (serious) game available on the net : JeuxDeMots. A propagation algorithm combining constraints and this network is presented.

5.1 A lexical network...

The structure of the lexical network we are building and using is composed of nodes and links between nodes, as it was initially introduced in the end of 1960s by Collins and Quillian (1) (developed by Sowa (13)), used in the small worlds by Gaume et al. (12), and more recently clarified by Polguère (2). A node of the network refers to a term (or a multiple word expression), usually in its canonical form (lemma). The links between nodes are typed and are interpreted as a possible relation holding between the two terms. Some of these relations correspond to lexical functions, some of which have been made explicit by Mel'cuk (18) and Polguère (2). It would have been desirable the network to contain all those lexical functions, but considering the principle of our software JeuxDeMots, it is not reasonably feasible. Indeed, some of these lexical functions are too much specialized; for example, Mel'cuk et al. (18) make the distinction between the Conversive, Antonym and Contrastive functions. They also consider refinements, with lexical functions characterized as *wider* or *more narrow*. JeuxDeMots being intended for users who are *simple Internet users*, and not necessarily experts in linguistics, such functions could have been badly interpreted by them. Furthermore, some of these functions are too poorly lexicalized, that is, very few terms possess occurrences of such relations; it is for example the case of the functions of Metaphor or Functioning with difficulty. More formally, a lexical network is a graph structure composed of nodes (vertices) and links.

- A node is a 3-tuple : `<name, type, weight>`
- A link is a 4-tuple : `<start-node, type, end-node, weight>`

The name is simply the string holding the term. The type is an encoding referring to the information holding by the node. For instance a node can be a term or a Part of Speech (POS) like :Noun, :Verb. The link type refer to the relation considered. A node weight is interpreted as a value referring to the frequency of usage of the term. The weight of a relation, similarly, refers to the strength of the relation.

JeuxDeMots possesses a predetermined list of relation types, and for now the players cannot add new types. Relation types fall into several categories:

- Lexical relations: synonymy, antonymy, expression, lexical family These types of relations are about vocabulary.
- Ontological relations: generic (hyperonymy), specific (hyponymy), part of (meronymy), whole of (holonymy) ... It is about relations concerning knowledge in objects of the world.
- Associative relations: free association, associated feeling, meaning It is rather about subjective and global knowledge; some of them can be considered as phrasal associations.
- Predicative relations: typical agent, typical patient ... They are about types of relation associated with a verb and the values of its arguments (in a very wide sense). Those are similar (if not identical) to semantic roles, which are of primary interest for us in this article.

The types of relation implemented in JeuxDeMots are thus of several natures, partially according to a distinction made by Schwab and Lafourcade (8): some of them are part of knowledge of the world (hyperonymy / hyponymy, for example), others concern linguistic knowledge (synonymy, antonymy, expression or lexical family, for example). Most players do not make this distinction which remains often vague for them. Here, the word *relation* has to be understood as an occurrence of relation, and not as a type of relation. Let us note that between two same terms, several relations of different types can exist.

5.2 ... a game for building it...

To ensure a system leading to quality and consistency of the base, it was decided that the validation of the relations anonymously given by a player should be made by other players, also anonymously. Practically, a relation is considered valid if it is given by at least one pair of players. This process of validation is similar to the one used by von Ahn et al. (14) for the indexation of images or more recently by Lieberman et al.(15) to collect common sense knowledge. As far as we know, this was never done in the field of the lexical networks. In Natural Language Processing, some other Web-based systems exist, such as Open Mind Word Expert (Mihalcea and Chklovski 22) that aims to create large sense tagged corpora with the help of Web users, or SemKey (Marchetti et al. 17) that exploits WordNet and Wikipedia in order to disambiguate lexical forms to refer to a concept, thus identifying a semantic keyword.

A game takes place between two players, in an asynchronous way, based on the concordance of their propositions. When a first player (A) begins a game, an instruction concerning a type of competence (synonyms, opposite, domains, ...) is displayed, as well as a term T randomly picked in a base of terms. This player A has then a limited time to answer by giving propositions which, to his mind, correspond to the instruction applied to the term T. The number of propositions which he can make is limited inducing players not just type anything as fast as possible, but to have to choose amongst all answers he can think of. The same

term, along the same instruction, is later proposed to another player B; the process is then identical. To increase the playful aspect, for any common answer in the propositions of both players, they receive a given number of points. The calculation of this number of points (as explained by Lafourcade and Joubert (16)) is crafted to induce both precision and recall in the feeding of the database. At the end of a game, propositions made by the two players are showed, as well as the intersection between these terms and the number of points they win.

According to the JeuxDeMots Web site, at the time of the writing of this paper, the lexical network contains more than 1100000 relations linking more than 230000 terms. Around 900000 games (with a mean of 1 minute per game) have been played corresponding to approximately 13000 hours (about 550 days) of cumulative play.

5.3 ... and a propagation algorithm

We devise a quite simple algorithm taking a set of constraints for input. A constraint here takes a form that is similar to an unweighted relation, that is to say a 3-tuple : `<start-node, type, end-node>`. One or both node can be a free variable with or without information of their POS. This form is directly mappable to relations in the lexical network. With the previous example, we have :

- `<X, :AGT, avocat>` and `<X, :AGT, dossier>`

Our algorithm is inspired by Page et al. (20) and partially by the LexRank algorithm (Bouklit and Lafourcade 6). The simple idea is to activate nodes of the lexical network that are involved in the constraints, and then to propagate activations along the network. As in practice, the entire network would be too large to be tractable, we reduce it only to the subset containing the first neighbours (at distance 1). All links are copied with the original weight when belonging to one of the constraints, or with weight equal to 1 otherwise. This propagation approach has been proven be convergent with a power iteration computing method. To be effective, the main hypothesis is that the set weight linking a node to its neighbour is considered as representing a probability distribution.

In the above example, an activation of 1 is inited to the node *avocat* and *dossier*. From *avocat*, all neighbours are copied with a link with a default value of 1, but those linked by a `:AGT` link. The process is similar for the node *dossier*. The spreading of the activation to a neighbour is done according to the ratio between this specific link weight. More precisely, the activation propagation from a node A to B is equal to the activation of A times the ratio of the weight of the link between A and B with the sum of the link weights from A). The output of the algorithm is *a set of activated terms*. In the above example, two terms are mostly activated: *plaider* and *étudier*. Furthermore, the proper specific meaning of *avocat>justice* (as *lawyer* and not *avocado*) is activated.

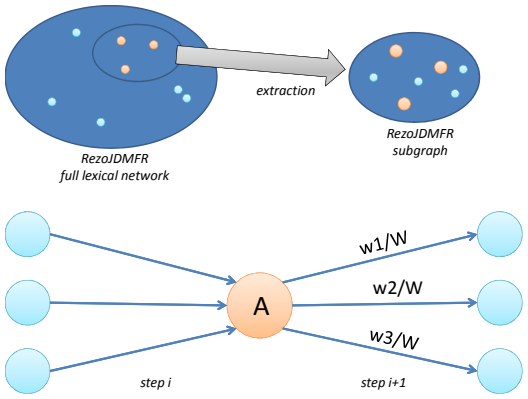

Fig. 4. Extraction of the lexical network subgraph and propagation of activation along nodes according to link weights.

6 Conclusion

The near absence of syntax–semantics interface for the Property Grammar framework is a serious impediment to its use for deep processing. Meanwhile, as a constituency-based MTS framework PG offers formal properties, which make it especially well-suited to address problems such as grammar error correction. In this paper we have started exploring the possibility to address the problem of the syntax–semantics interface through the pairing with a semantic network. We have shown that the linguistic knowledge contained within PG constraints, together with a deep phrase structure, allows for the construction of detailed semantic information about a—possibly ill—sentence. Such semantic information could not be gathered by compositional means only. We have also shown how to use that information to explore a semantic network and find suitable lexical items to complete a sentence missing words, with an emphasis on missing predicates.

References

A., C., M.R., Q.: Retrieval time from semantic memory. Journal of verbal learning and verbal behaviour 8(2), 240–248 (1969)

A., P.: Structural properties of lexical systems: Monolingual and multilingual perspectives. In: Proc. of the Workshop on Multilingual Language Resources and Interoperability (COLING/ACL 2006) (2006)

Bès, G., Blache, P.: Propriétés et analyse d'un langage. In: Proc. of the 1999 Conf. on Traitement Automatique du Langage Naturel (TALN'99) (1999)

Blache, P.: Les Grammaires de Propriétés : des contraintes pour le traitement automatique des langues naturelles. Hermès Sciences (2001)

Blache, P.: Property Grammars: A Fully Constraint-based Theory. In: Christiansen, H., Skadhauge, P.R., Villadsen, J. (eds.) Constraint Solving and Language Processing, LNAI, vol. 3438. Springer (2005)
Bouklit, M., Lafourcade, M.: Propagation de signatures lexicales dans le graphe du web. In: Proc. of RFIA'2006, Tours, France, (2006)
Chauché, J.: Un outil multidimensionnel de l'analyse du discours. In: Proc. of the 10th Int'l Conf. on Computational Linguistics and 22nd annual meeting on Association for Computational Linguistics. pp. 11–15. ACL (1984)
D., S., M., L.: Modelling, detection and exploitation of lexical functions for analysis. ECTI Journal 2, 97–108 (2009)
Dahl, V., Gu, B.: On semantically constrained property grammars. In: Constraints and Language Processing. p. 20 (2008)
Duchier, D., Dao, T.B.H., Parmentier, Y., Lesaint, W.: Property Grammar Parsing Seen as a Constraint Optimization Problem. In: Proceedings of the 15th Intl Conference on Formal Grammar (FG 2010) (2010)
Duchier, D. and Prost, J.-P. and Dao, T.-B.-H.: A model-theoretic framework for grammaticality judgements. In: Proc. of FG'09. Lecture Notes in Artificial Intelligence, vol. 5591. Springer (2009)
Gaume B., D.K., M., V.: Semantic associations and confluences in paradigmatic networks. In: Vanhove, M. (ed.) Typologie des rapprochements sémantiques (2007)
J., S.: Semantic networks. Encyclopedia of Artificial Intelligence. edited by S.C. Shapiro, Wiley, New York (1992)
vonAhn L., L., D.: Labelling images with a computer game. In: Proc. of ACM Conf. on Human Factors in Computing Systems (CHI) (2004)
Lieberman H., S.D., A., T.: Common consensus: a web-based game for collecting commonsense goals. In: Proc. of Int'l Conf. on Intelligent User Interfaces (IUI'07) (2007)
M., L., A., J.: Détermination des sens d'usage dans un réseau lexical construit à l'aide d'un jeu en ligne. In: Proc. of the Conférence sur le Traitement Automatique des Langues Naturelles (TALN'08) (2008)
Marchetti A., Tesconi M., R.F.R.M., S., M.: Semkey: A semantic collaborative tagging system. In: Proc. of SemKey: A Semantic Collaborative Tagging System (2007)
Mel'cuk I.A., Clas A., P.A.: Introduction à la lexicologie explicative et combinatoire. Ed. Duculot AUPELF-UREF (1995)
Messiant, C.: A Subcategorization Acquisition System for French Verbs. In: Proc. of the ACL-08: HLT Student Research Workshop. pp. 55–60. ACL (2008)
Page L., S. Brin, R.M., Winograd, T.: The PageRank Citation Ranking : Bringing Order to the Web. Technical report, Stanford University (1998)
Pullum, G., Scholz, B.: On the Distinction Between Model-Theoretic and Generative-Enumerative Syntactic Frameworks. In: de Groote, P., Morrill, G., Rétoré, C. (eds.) Logical Aspects of Computational Linguistics: 4th International Conference. pp. 17–43. No. 2099 in LNAI, Springer Verlag (2001)
R., M., T., C.: Open mind word expert: Creating large annotated data collections with web users' help. In: Proc. of the EACL 2003 Workshop on Linguistically Annotated Corpora (LINC 2003) (2003)
VanRullen, T.: Vers une analyse syntaxique à granularité variable. Ph.D. thesis, Université de Provence, Informatique (2005)
Wagner, J., Foster, J., van Genabith, J.: Judging grammaticality: Experiments in sentence classification. CALICO Journal 26(3), 474–490 (2009)
Wong, S., Dras, M.: Parser Features for Sentence Grammaticality Classification. In: Australasian Language Technology Association Workshop 2010. p. 67 (2011)