Modelling Global Phenomena with Extended Local Constraints

Patrick McCrae Kilian A. Foth Wolfgang Menzel
{mccrae,foth,menzel}@informatik.uni-hamburg.de

Natural Language Systems Group
Department of Informatics
Hamburg University
D-22527 Hamburg
Germany

Abstract. Weighted constraint dependency systems (WCDG) perform in the antagonism between constraint expressivity and processing limitations. While for reasons of efficiency most implementations constrain themselves to the evaluation of localised unary and binary constraints, a whole range of modelling tasks require access to supra-local or global properties. In constraint dependency grammars, however, these can only be accessed via higher arity constraints. Typical examples from syntax include tests for verb valency or active/passive voice.
In this paper, we illustrate how two additional predicates (`is` and `has`) permit to model global and supra-local properties within the boundaries of a binary constraint syntax. With this extension, higher arity constraints can be cast as a single binary constraint calling unary ancillary constraints cascadingly or recursively. We demonstrate that our method is not limited to modelling strictly adjacent dependencies, but that it can be applied to an arbitrary set of dependencies connected in a dependency tree. In making global properties accessible within an otherwise localised formalism, this work provides significant expressivity enhancements in constraint-based dependency systems.

1 Global Phenomena in Dependency Grammar

Weighted constraint dependency grammar is a formalism which attains its global solution to a constraint satisfaction problem from searching a solution space containing ranked candidates obtained from the optimisation of local constraint evaluations. Given sufficient time, a complete search will find the global optimum as defined by the given set of constraints. Since the complexity of constraint evaluation grows exponentially with the highest constraint arity in the grammar, real-world applications are often restricted in their arity by performance considerations. Constraint-based systems also face a challenge arising from the very nature of their formalism: for unification-based parsing approaches such as HPSG or LFG, propagating feature information through the solution structure constitutes a fundamental principle, but for constraint-based dependency systems with

an inherently localised view, this process presents a considerable challenge. In the following we refer to any property which cannot be tested for by a single localised constraint as *supra-local*. Properties requiring knowledge of the entire solution structure we refer to as *global*. Both the original CDG [Maruyama, 1990] and the later extension WCDG originally supported only unary and binary constraints; in comparison, the similar XDG [Duchier, 1999] supports supra-local constraints by running on top of a general constraint programming system (Oz). In this paper we will show how the extension of the WCDG formalism by the two predicates `is` and `has` opens up new modelling pathways for constraint-based systems that—in combination with certain solution procedures—permit access to complex feature information as if it were propagated along dependency edges. Our work provides solid grounds for challenging the traditional view onto constraint-based formalisms as adopting a strictly localised view. Motivated by a range of German language modelling challenges, we present four applications of the new predicates that illustrate how they can be employed to re-formulate constraints whose complexity exceeds that of the classical weighted constraint dependency formalism based on unary and binary constraints.
This paper is structured as follows: In the following section we provide an introduction into the classical expressivity of the WCDG formalism. Section 3 outlines our extensions to that formalism based on modelling examples. The examples in Section 3 are ordered by increasing constraint expressivity as required by the respective modelling scenarios. Section 4 summarises our key findings, and Section 5 provides an outlook onto future directions for our research.

`X↑from`	linear position of the regent of edge X
`X↓word`	word form of the dependent of edge X
`X↑case`	'case' feature of the regent of edge X
`X.label`	dependency label of edge X
`&  |  ->  <->  ~`	logical operators: *and*, *or*, *implication*, *biimplication*, *not*
`<  >  <=  >=  !=  =`	numerical logical operators: *less*, *greater*, *equal*, *unequal*

Table 1. Operators in WCDG

2 Expressivity of WCDG

The version of dependency grammar which we use in this discussion is WCDG [Menzel and Schröder, 1998], an implementation that allows the definition of weighted constraints on dependency structures across multiple levels. Taken together, all constraints define a grammar that assigns every possible dependency tree a score (essentially, the product of the scores of all violated constraints), thus defining a total ordering over all possible structures, where the highest-scoring structure is taken as the preferred one.
Grammar rules take the form of unary or binary all-quantified logical formulas describing properties that a dependency analysis should have. The most important operators available to the grammar writer are shown in Table 1;

full details can be found in [Schulz et al., 2005]. By combining these operators, logical conditions can be expressed that can be tested on one or two edges in a dependency structure at a time. For instance, the following real-world example penalises right extraposition of relative clauses in German main clauses:

```
{X:SYN,Y:SYN} : 'Extraposition über das Verb' : proj : 0.5 :
  X.label = SUBJ & Y.label = REL & X/ & Y\ & X↓from = Y↑from
  -> Y↓from < X↑from;
```

This constraint can be read as follows: When a relative clause (`Y.label=REL`) modifies a word (`Y↑from = X↓from`) that is a subject (`X.label=SUBJ`), and the relative clause is right-modifying (`Y\`) while the subject is left-modifying (`X/`), then the relative clause itself (`Y↓from`) must occur to the left (`<`) of the finite verb (`X↑from`). With constraint weights ranging between 0 (hard) and 1 (soft) this constraint's score of 0.5 indicates that it describes a dispreference rather than an outright prohibition. This accurately models the observation that while relative clauses can be extraposed out of their topological field, a position within the field of the antecedent is preferred:

> "Ein Flugzeug, das drei Passagiere an Bord hatte, ist im Meer abgestürzt."
> ?"Ein Flugzeug ist im Meer abgestürzt, das drei Passagiere an Bord hatte."
> *(A plane carrying three passengers has crashed into the sea.)*

While the formulation of grammar rules as defeasible, declarative constraints often seems odd to grammarians accustomed to constituent descriptions or generative rules, it is a flexible and highly expressive means of describing linguistic phenomena, particularly when a non-prescriptive grammar is intended. (For instance, a grammar of modern English might employ essentially the same constraint, but assign it a much stricter score.) In fact, as long as a configuration of not more than two dependency edges suffices to describe a phenomenon, WCDG can express any constraint on dependency structures that are formally definable.

3 Extending Local Constraints to Global Phenomena

3.1 Supra-local Constraints

The foremost linguistic phenomenon that cannot be expressed as a local condition is valency. The valency requirement of a word is a local phenomenon in the sense that a single dependency suffices to prove that it is satisfied. Still, it cannot be tested for by all-quantified binary constraints; because any of the other words in a sentence might be the required dependant, *all* dependency edges would have to be checked. Maruyama's solution [Maruyama, 1990] was to establish a second, sparse tree structure that mirrors syntactic valency dependencies as inverse dependencies on an auxiliary level. The mirror condition can be ensured with a binary constraint that couples both levels, and the valency itself requires only a unary constraint on the auxiliary level.

This is a viable solution, but leads to a profusion of auxiliary structures when more than one type of valency must be required, as is often the case with realistic verb forms. An alternative solution is to introduce new operators into the constraint language that can express the existence condition directly. This allows a more readable formulation of valency constraints and avoids auxiliary structures that exist for technical reasons only, but has an important consequence: a constraint which uses such a non-local operator cannot be evaluated in isolation anymore. Rather than just one or two at a time, the entire set of dependency edges in a solution candidate must be known to decide whether or not it satisfies such a supra-local constraint. This means that a supra-local constraint cannot safely be evaluated on partial structures, as is done during a propagation algorithm strategy or a best-first search. The best one can do is to delay the use of these constraints until a complete solution candidate is found and only then to apply the entire grammar. This method resembles the two-step re-ranking technique that is becoming common for other complete search algorithms [Charniak and Johnson, 2005].

Such an approach may be viable as long as only a few of the constraints in a grammar require supra-local conditions, but for a grammar that relies on them more heavily, the first step could become too unfocused to deliver good results. An alternative solution is to use algorithms that do have access to the entire solution candidate from the beginning. WCDG does provide such algorithms; in fact they have been proven to be the best solution strategy [Foth et al., 2000]. The basic strategy is to construct an initial analysis from the (unarily) best-ranked dependencies and then systematically to exchange those dependencies that violate constraints for others which satisfy them. This approach has proved to be workable despite its heuristic nature, and, in fact, is more successful than the (theoretically) complete but infeasible search, while preserving the anytime property of the filtering algorithms. This has prompted the implementation of several extended operators that rely on supra-local properties, and work well together with transformational search [Foth, 2007].

The foremost of these new operators in WCDG is called `has`. In its simplest form, it expresses the condition that a particular word is modified by at least one dependency edge that bears a specific label. This allows conditions like the following to be expressed:

```
X↓cat = VVFIN -> has(X↓id, SUBJ)   (Finite verbs need subjects.)
X↓cat = VVFIN & X↓transitive = yes -> has(X↓id, OBJA)
                                   (Transitive verbs need objects.)
```

Although the obvious application of this operator is to enforce verb valency conditions, it lends itself to many related uses. For instance, German infinitive constructions can occur in the form of an infinitive with a particular marker word (of the category `PTKZU`), or as a special verb form that incorporates this marker (category `VVIZU`). Assuming that an external marker always bears the special-purpose label `ZU`, the `has` operator allows this condition to be expressed concisely:

```
X↓cat = VVIZU | (X↓cat = VVINF & has(X↓id, ZU))
```
(X↓ *is a valid infinitival construction.*)

The WCDG engine deduces the supra-local nature of a formula automatically by scanning its body and applies such constraints only if a complete solution candidate is available. Thus, conditions are still expressed over single edges or pairs of edges at a time, but during evaluation they can also examine additional neighbouring edges as required.

While the `has` operator expresses conditions on the dependents of a word, the similar `is` operator tests the label of the dependency edge above a given word. For instance, German main clauses generally place exactly one constituent in front of the final verb. This can be expressed by a constraint that forbids two dependencies modifying the same verb from the left. However, this condition only holds in main clauses (in which the verb itself is labelled as `S`), but not for subclauses or relative clauses (labelled e.g. `NEB` or `REL`). Therefore, three edges would have to be tested to detect such an illegal configuration: the two right-modifying depencies under the verb and the edge directly above. This would require a ternary constraint, which WCDG does not support. With the supra-local `is` operator, however, a single binary constraint suffices:

```
{X/SYN/\Y/SYN} : Vorfeld : 0.1 :
  X↑cat = VVFIN -> ~is(X↑id, S);
```

In fact, this constraint is considerably faster to evaluate than an all-quantified ternary constraint would be, because WCDG effectively has to check one additional edge (the one above the finite verb), and that only when the premise of the constraint actually holds. Thus, it allows for easier grammar development and more efficient evaluation than a grammar limited to strictly local constraints.

3.2 Recursive Tree Traversal

One limitation of the supra-local operators `is` and `has` described so far is that they operate only on direct neighbours of the dependency edges to which they are applied. This is often sufficient, but there are phenomena which require the presence of structurally more distant features. For instance, a subclause should be analysed as a relative clause (`REL` instead of `NEB`) exactly if it is marked by the presence of a relative pronoun, but this pronoun does not always modify the finite verb directly:

> "Es soll eine Art Frühwarnsystem eingerichtet werden, in dessen Zentrum der IWF steht."
> (*An early-warning system is planned whose center will be constituted by the IWF.*)

Similar cases of remote markers abound in German: the conjunction *sondern* is only used for phrases containing a negation somewhere, a genitive modifier must contain at least one overt genitive form, etc. To check such conditions, it is

necessary to extend the semantics of the supra-local operators so that optionally they can also find indirect dependants or regents. In such cases it is useful to restrict the extended search in some way, both for operational and for linguistic reasons. For instance, when a subclause is modified by a nested relative clause, the subclause itself should not be labelled `REL`, even though the corresponding dependency subtree contains a relative pronoun further down. Similarly, in co-ordinated sentences the finite verb is labelled as `KON` (in asyndetic co-ordination) or `CJ` (in normal co-ordination) rather than `S` or `NEB`, so that even a lookup via `is` cannot determine whether main-clause or subclause ordering should be enforced; what counts is the label of the topmost finite verb in a co-ordination, which can be several edges apart.

Therefore, the notion of 'scope' has been implemented for the extended versions of the non-local operators: when used with four arguments, the search is extended across a specific set of labels, i.e. those which are subsumed by a particular pseudo-label in a special-purpose hierarchy. For instance, the actual test for sentence type in the Vorfeld constraint is closer to the following version:

`is(X↑id, S, Label, Konjunkt)` (`X↑` *is eventually labelled* `S`)

where `Konjunkt` subsumes both `KON` and `CJ` in the hierarchy 'Label'.

This construct effectively ascends the tree from a finite verb until a label other than `KON` or `CJ` is found, and compares this label to the main-clause marker `S`. The `has` operator has been extended in the corresponding way; for instance, it can be programmed to descend into a sentence labelled `REL` to detect a relative pronoun, but only until another subclause indicator such as `REL`, `NEB` or `S` intervenes. This use of a label-delimited semi-global search resembles the notion of *barriers* in Government and Binding theory [Chomsky, 1986], but it does not claim to be a fundamental principle. Indeed, by varying the set of labels to traverse, it can be restricted more or less; for instance, it can operate only upon an NP, or upon the entire tree structure.

3.3 Localised Ancillary Constraints

The syntax for the `is` and `has` predicates introduced so far permits to test for static attributes of the edges above or below the dependency under consideration. A useful extension to the concepts of `is` and `has` therefore is to include a check for the most general edge property expressible: the satisfaction of an arbitrary constraint. Since `is` and `has` are evaluated in the context of a normal constraint, we refer to their argument constraint as *ancillary constraint*. To motivate this extension linguistically, consider thematic role assignment in German non-modal perfect tense active sentences:

Der Mann [AGENT] hat die Frau mit dem Fernrohr gesehen.
(*The man* [AGENT] *has seen the woman with the telescope.*)

In all of the following examples we assume the full verb to be agentive. `SUBJ` and `AGENT` dependencies then originate from the same node in the constraint

net. Non-modal perfect tense active in German is a composite tense formed by a finite auxillary in combination with a full verb's past participle. In constraint terms, this tense can be characterised by a dependency with the following properties: An `AUX` edge (`X.label=AUX`) links a finite auxilliary verb form (`X↑cat=VAFIN`) of *haben* or *sein* (`X↑base=haben | X↑base=sein`) as regent with a full verb's past participle (`X↓cat=VVPP`) as dependant.
Figure 1 illustrates that constraining the origin of the `AGENT` dependency (orange) to the origin of the `SUBJ` dependency (blue) in a non-modal perfect passive sentence requires a ternary constraint involving the `SUBJ`, `AGENT` and `AUX` (green) edges. Moreover, formulating this constraint requires to impose restrictions on the `AUX` edge as well as on the nodes linked by it.
In requiring satisfaction of an ancillary constraint via `is` or `has`, the origin of the `AGENT` dependency in German perfect tense active sentences can elegantly be formulated as follows: A `SUBJ` edge (`X.label=SUBJ`) meeting with an `AUX` edge that marks a perfect active sentence (`has(X↑id, 'Detect perfect tense active')`) must have an edge originating from its bottom node (`X↓id = Y@id`) which bears the label `AGENT` (`Y.label=AGENT`). It is this use of `has` in combination with the ancillary constraint that allows us to express a genuinely ternary supra-local relation as a WCDG-licensed binary constraint.
The ancillary constraint to be satisfied by the edge meeting the `SUBJ` dependency enforces exactly the set of properties previously identified for the detection of a perfect tense active sentence:

```
{X:SYN} : 'Detect perfect tense active' : ancillary : 1 :
  X.label = AUX
  & X↑cat = VAFIN
  & (X↑base = haben | X↑base = sein)
  & X↓cat = VVPP;
```

By employing the ancillary constraint as argument to `has`, we have effectively extended the scope of properties accessible on a neighbouring dependency—from access to a single static edge property to the full range of edge and node properties available. Constraint expressivity is enhanced because we can now create general custom predicates that neighbouring edges need to fulfill. Clearly, the conjunction of features `X↑cat=VAFIN & (X↑base=haben | X↑base=sein) & X↓cat=VVPP` was intractable with the static arguments to `is` or `has` presented in the previous sections.
The elegance of this approach lies in the fact that ancillary constraints of arbitrary complexity can now be employed as re-usable functional blocks to perform checks for linguistically intuitive, yet formally complex properties over and over again. Notable from a performance point of view is, that the WCDG implementation is such that, once an ancillary constraint has been evaluated for a given edge, its result will be cached and afterwards is available for repeated use at no extra cost computationally.

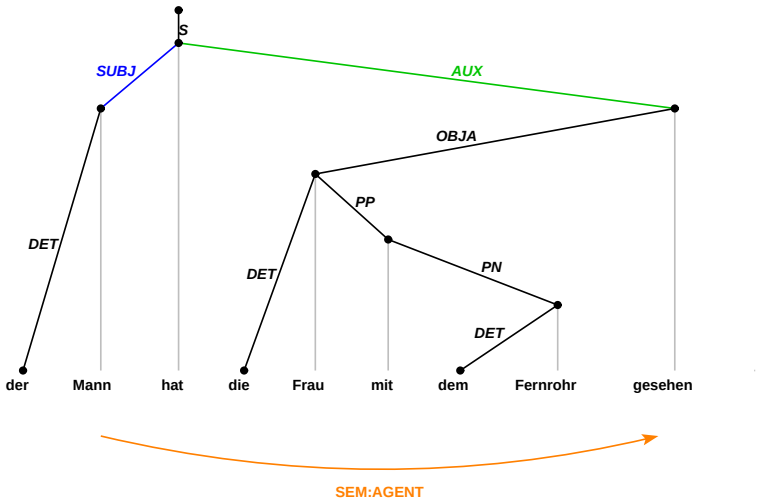

Fig. 1. AGENT assignment in German perfect tense active sentences

3.4 Cascading and Recursive Ancillary Constraints

The ancillary constraints presented so far are localised unary constraints—and as such provide full access to the properties of the next-neighbour edges above and below a given dependency in the syntax tree. As we will now illustrate, the syntax-semantic interface exhibits phenomena the modelling of which requires even higher expressivity than is provided by the extended localised unary ancillary constraints. We proceed to describe an additional expressivity enhancement that utilises cascading and recursive calls to ancillary constraints. This enables us to model properties spanning across arbitrarily large sections of the dependency tree, e.g. global properties, with just binary constraints.

As an example consider `AGENT` thematic role assignment in German passive sentences. The `AGENT` in a German passive sentence typically is embedded as the PP filler noun (`X.label=PN`) in a *von*-PP (`X.label=PP & X↓word=von`) which modifies the past participle of a full verb(`X↑cat=VVPP`) (see Figure 2).

> Der Mann wird von der Frau [AGENT] mit dem Fernrohr gesehen.
> (*The man is being seen by the woman* [AGENT] *with the telescope.*)

The full-verb past participle, in turn, must be correctly embedded in the lowest-lying `AUX` dependency in order for the sentence to be in passive voice. We can therefore formulate the constraint on the origin of the `AGENT` dependency in German passive sentences with the following cascade of ancillary constraints (colours refer to the highlights in Figure 2):

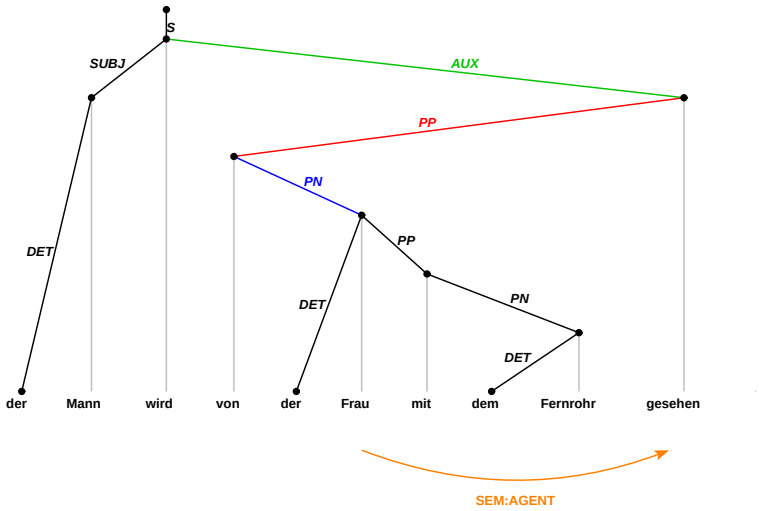

Fig. 2. AGENT assignment in German passives

Binary invocation constraint
For the pair of edges X (orange) and Y (blue) which share the same origin node (`X↓id = Y↓id`) we demand: If Y is a `PN` edge (blue) and the edge above it (red) satisfies the ancillary constraint `'Detect full-verb modifying von-PP in passive'`, then X (orange) must be an `AGENT` dependency.

```
{X:SEM,Y:SYN} :
  X↓from = Y↓from & Y.label = PN
  & is(Y↑id, 'Detect full-verb modifying von-PP in passive')
  -> X.label = AGENT;
```

Ancillary constraint #1: `'Detect full-verb modifying von-PP in passive'`.

The red edge above `PN` (blue) must be a full-verb modifying *von*-PP. This is tested for by ancillary constraint #3. The edge above the red edge must be the lowest-lying `AUX` edge in a passive construction (green), which is tested for in ancillary constraint #2.

```
is(X↓id, 'Detect full-verb modifying von-PP')
& is(X↑id, 'Detect passive bottom-up');
```

Ancillary constraint #2: `'Detect passive bottom-up'`.

The edge above the full-verb modifying `PP` must be a passive-marking `AUX` edge. Passive sentences are identified based on their lowest-lying `AUX` edge (green) which connects a past participle dependant with its auxilliary regent of base form *werden*. The regent's category depends on tense.

```
X.label = AUX
& X↓cat = VVPP
& ~has(X↓id, AUX)
& (X↑cat = VAFIN                          // Pres, Simple Past
   | (X↑cat = VAPP & is(X↑id, AUX) )      // Perfs, FutII, SubjII
   | (X↑cat = VAINF & is(X↑id, AUX) ) )   // FutI, SubjI
& X↑base = werden;
```

Ancillary constraint #3: `'Detect full-verb modifying von-PP'`.

A PP is of relevance to `AGENT`-assignment in a passive constructions if it contains the preposition *von* and attaches to the full verb's past participle.

```
X.label= PP & X↑cat = VVPP & X↓word = von;
```

Again, use of an ancillary constraint permits us to express in a binary constraint a condition which otherwise would have required a quarternary constraint construction relating the `PN`, `PP`, `AUX`, and `AGENT` dependencies.

A related, though again slightly more complex modelling task is to constrain the origin of the `AGENT` dependency in German active sentences. Due to the large number of structurally diverse active constructions in German it can be more convenient to model an active voice sentence as a sentence which is not in passive voice.[1] As mentioned above, German passives can be identified based on their lowest-lying `AUX` edge. Since the actual location of this edge depends on tense and mode, the constraint for its detection needs to be flexible. We employ a constraint which moves down the dependency tree by recursively invoking itself until it either finds an `AUX` edge satisfying the bottom-up criteria for passive detection or until it cannot descend further and fails altogether. The `AGENT` dependency in an active voice sentence may originate from the origin of the `SUBJ` dependency, while in a passive voice sentence it originates from the origin of the `PN` dependency contained in a full-verb modifying PP. Note that these conditions include the global properties *active* and *passive* voice. We can now conveniently formulate this complex requirement in the following recursive ancillary constraint invocation: Given an `AGENT` dependency, it originates either from a `SUBJ` edge in an active sentence or from a `PN` edge in a full-verb modifying *von*-PP in a passive construction.

```
X.label = AGENT ->
  (Y.label = SUBJ & ~has( Y↑id, 'Detect passive sentence top-down')) |
  (Y.label = PN
   & is(Y↑id, 'Detect full-verb modifying von-PP in passive'));
```

While the detection procedure for the full-verb modifying *von*-PP in a passive construction has been outlined above, the detection of the active construction merits further explanation. Starting from the `SUBJ` edge (blue) in Figure 3, the ancillary constraint first checks the green `OBJA` edge (green) for satisfaction of the ancillary

[1] This modelling decision may require justification beyond the scope of this paper. Suffice it here to say that replacing the indirect detection of active voice by a direct detection has no impact on our line of argument. The ancillary constraints merely need to be re-formulated to detect the structural features of an active voice sentence.

constraint `'Detect passive bottom-up'`. Since this is unsuccessful, it then continues to descend down the right hand-side of the dependency tree, progressing edge by edge, recursively re-invoking itself until it either finds an `AUX` edge satisfying the ancillary constraint and terminates, or until no further alternatives are available and it fails.

```
{X:SYN} : 'Detect passive top-down' : ancillary : 1 :
  X.label = AUX
  & (is( X↓id, 'Detect passive bottom-up')
     | has( X↓id, 'Detect passive top-down'));
```

This formulation is an expressive extension to the recursive tree traversal introduced in Section 3.2.

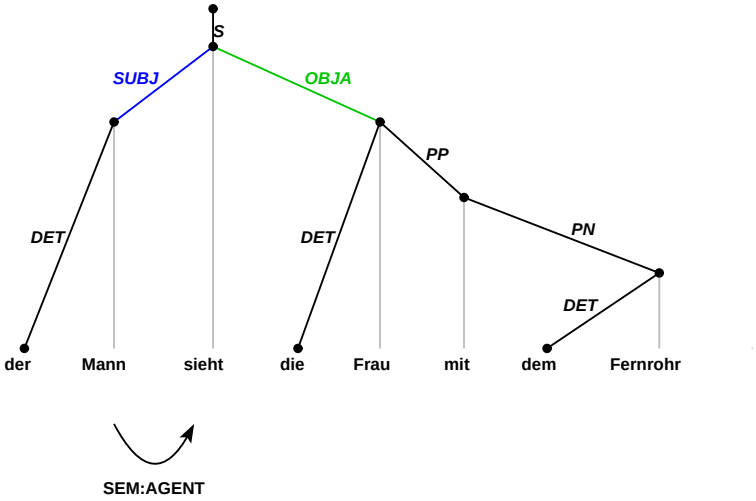

Fig. 3. AGENT assignment based on active/passive detection in German

The increase in constraint expressivity required in this modelling scenario arises from the fact that the dependencies constrained are farther apart in the dependency tree and thus are not contiguous anymore. So, although our approach for extending access to edge and node properties of neighbouring dependency edges is based on `is` and `has`, it is by no means limited in applicability to neighbouring edges. Before, supra-local properties would have needed to be tested for by a higher arity constraint, which was unavailable in WCDG's formalism. Now, such a higher arity constraint can be re-formulated as a suitably expressive binary constraint operating on a contiguous dependency structure that contains all edges we wish to predicate. Since an ancillary constraint can only extend access to one neighbouring dependency above or below, there is a linear relationship between the number of invocations to an ancillary constraint and the distance between the considered edges in the dependency tree.

4 Conclusions

In this paper we have outlined the gain in constraint expressivity achieved with the introduction of two additional predicates `is` and `has`. We showed how these predicates

extend constraint access in the dependency tree and thus open up a path to the effective and efficient handling of higher arity constraints within the formal and operational limitations of the WCDG formalism. Motivated by examples from the syntax-semantics interface, we illustrated that the consecutive extension of the predicate syntax for `is` and `has` in combination with cascading and recursive invocations to ancillary constraints produces a significant increase in constraint expressivity. Most notably, we have demonstrated how global syntactic properties such as active or passive voice can be made accessible within a binary constraint dependency formulation.

5 Future work

Our work so far has focused on implementations involving unary ancillary constraints. With few changes the WCDG formalism can be extended to support the evaluation of binary ancillary constraints as well. A systematic investigation into the effects of this is pending.
From a theoretical point of view, a formal analysis of the expressivity enhancements achieved with `is` and `has` appears challenging and rewarding. While we have focused on the use of `is` and `has` to solve specific modelling tasks, we conjecture that the full expressive potential resulting from the use of these predicates in combination with ancillary constraints has not yet been exhausted.

6 Acknowledgements

Parts of this research were supported by the CINACS graduate research project funded by the German Research Foundation (DFG). We would also like to express our sincere thanks to the four anonymous reviewers for their detailed and constructive feedback on an earlier version of this paper.

References

[Charniak and Johnson, 2005] Eugene Charniak and Mark Johnson. 2005. Coarse-to-Fine n-Best Parsing and MaxEnt Discriminative Reranking. In *Proc. ACL 2005*, pages 173–180, Ann Arbor, Michigan, June. Assoc. for Computational Linguistics.

[Chomsky, 1986] Noam Chomsky. 1986. *Barriers*. MIT Press.

[Duchier, 1999] Denys Duchier. 1999. Axiomatizing dependency parsing using set constraints. In *Proceedings of MOL6*, pages 115–126, Orlando/USA.

[Foth et al., 2000] Kilian A. Foth, Wolfgang Menzel, and Ingo Schröder. 2000. A Transformation-based Parsing Technique with Anytime Properties. In *4th Int. Workshop on Parsing Technologies, IWPT-2000*, pages 89–100.

[Foth, 2007] Kilian A. Foth. 2007. *Hybrid Methods of Natural Language Analysis*. Ph.D. thesis, Universität Hamburg.

[Maruyama, 1990] Hiroshi Maruyama. 1990. Structural disambiguation with constraint propagation. In *Proc. 28th Annual Meeting of the ACL (ACL-90)*, pages 31–38, Pittsburgh, PA.

[Menzel and Schröder, 1998] Wolfgang Menzel and Ingo Schröder. 1998. Decision procedures for dependency parsing using graded constraints. In Sylvain Kahane and Alain Polguère, editors, *Proc. Coling-ACL Workshop on Processing of Dependency-based Grammars*, pages 78–87, Montreal, Canada.

[Schulz et al., 2005] Michael Schulz, Stefan Hamerich, Ingo Schröder, Kilian Foth, and Tomas By, 2005. *[X]CDG User Guide*. Natural Language Systems Group, Hamburg University, Hamburg, Germany.