VP Ellipsis by Tree Surgery

Katrin Erk and **Alexander Koller**
erk@ps.uni-sb.de / koller@coli.uni-sb.de
Saarland University, Saarbrücken, Germany

Abstract

We present *jigsaw parallelism constraints*, a flexible formal tool for replacing parts of trees with other trees. Jigsaw constraints extend the Constraint Language for Lambda Structures, a language used in underspecified semantics to declaratively describe scope, ellipsis, and their interaction, and can be used to improve the coverage of ellipsese represented by CLLS.

Keywords: tree descriptions, ellipsis, underspecified semantics

1 Introduction

In this paper, we define *jigsaw parallelism constraints*, a versatile tool for replacing parts of trees and λ-terms. Jigsaw constraints are a generalization of the *parallelism constraints* defined in the framework of the Constraint Language for Lambda Structures (CLLS, Egg et al. 2001). CLLS is a logical language interpreted over tree-like structures which can be applied to underspecified semantics.

CLLS has been used for declarative underspecified descriptions of the meaning of scope, simple anaphora, reinterpretation in lexical semantics (Koller et al. 2000), and simple ellipses. Inspired by the semantic approaches to ellipsis based on higher-order logic (Dalrymple et al. 1991; Crouch 1995; Gardent and Kohlhase 1996), the ellipsis theory in the CLLS framework operates on a λ-term representing the meaning of the sentence, and then replaces the meaning of the parallel element in the source sentence by the meaning of the one in the target sentence. Unlike the HOU-based approaches, this replacement applies directly to specific nodes in a tree encoding of the term; different occurrences of subterms are automatically kept apart.

While this approach can account for quite a few examples from the literature, including interactions with anaphora and scope, there are ellipses which are simple for other approaches, but cannot be handled in CLLS. The second contribution of this paper is to show how jigsaw parallelism can handle these examples. This is possible because jigsaw parallelism allows a much finer specification of which parts of the tree should be replaced.

The paper is structured as follows. In Section 2, we give a very brief introduction to the framework of CLLS. Section 3 sketches how ellipses are usually analyzed in CLLS, and presents some examples that CLLS cannot handle. Then we define jigsaw parallelism in Section 4 and show how to account for the problematic sentences in Section 5. Section 6 concludes the paper, presents some thoughts on the processing of jigsaw parallelism constraints, and sketches an application of the tree surgery made possible by jigsaw parallelism to TAG.

We can only sketch the definitions and lemmas in this abstract. The exact definitions and proofs of the claims we make below will be made available in an extended version on the web (Erk and Koller 2001).

2 The Constraint Language for Lambda Structures

The jigsaw parallelism constraints we want to define below are a conservative extension of the Constraint Language for Lambda Structures (CLLS) and generalize the *parallelism constraints* provided there. We give the briefest possible overview of CLLS; for a more careful introduction (and clean definitions), see (Egg et al. 2001).

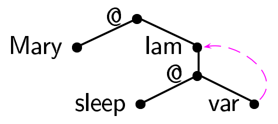


Figure 1: Lambda structure representing the λ-term $\mathsf{Mary}(\lambda x.\mathsf{sleep}(x))$.

CLLS is a language used for the partial description of *lambda structures*, which can be used to encode λ-terms. Lambda structures are ordinary trees which have been enriched by two partial functions λ (which models variable binding) and ante (which models anaphoric reference). An example, encoding the λ-term $\mathsf{Mary}(\lambda x.\mathsf{sleep}(x))$, is shown in Fig. 1. Application is represented as the binary label @. Abstraction and bound variables are represented using the labels lam and var, and the λ-binding function (indicated by the dashed arrow) is used to indicate which

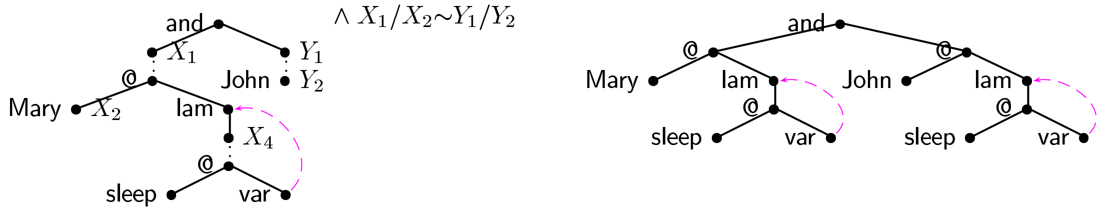


Figure 2: A parallelism constraint for (1) and a lambda structure that satisfies it.

variable is bound by which binder.

The syntax of CLLS is defined as follows; X, Y, etc. are variables which denote nodes in a lambda structure; A and B are explained below.

$$\begin{array}{lll} \varphi & ::= & X{:}f(X_1,\ldots,X_n) \mid X\triangleleft^* Y \mid A \sim B \\ & \mid & \lambda(X){=}Y \mid \mathsf{ante}(X) = Y \mid \varphi \wedge \varphi' \end{array}$$

That is, a CLLS formula or *constraint* is a conjunction of atomic literals; it is satisfied by a lambda structure and a variable assignment iff all these literals are satisfied in the following sense. A *labeling literal* $X{:}f(X_1,\ldots,X_n)$ is satisfied iff X denotes a node with label f and children that are denoted by $X_1,\ldots,X_n$. A *dominance literal* $X\triangleleft^* Y$ is satisfied iff X denotes a (reflexive, transitive) ancestor of Y in the lambda structure. The *binding literals* $\lambda(X){=}Y$ and $\mathsf{ante}(X){=}Y$ are satisfied iff the respective binding functions map the denotation of X to the denotation of Y.

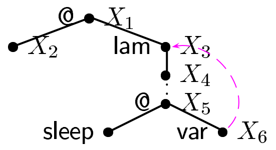


Figure 3: A CLLS constraint satisfied by the structure in Fig. 1.

We usually draw constraints as *constraint graphs*, as in Fig. 3. The nodes in this graph stand for variables in a constraint and the edges and labels represent different types of literals. This particular graph represents a constraint that starts $X_1{:}@(X_2, X_3) \wedge X_4\triangleleft^* X_5 \wedge \lambda(X_6){=}X_3 \wedge \ldots$ and, incidentally, is satisfied by the lambda structure in Fig. 1.

The most complex, and for this paper the most important, literal is *parallelism* $A \sim B$. In the original definition, A and B are terms of the form X/Y. This term denotes a *segment* of the lambda structure: a pair u/v of a *root* u and a *hole* v. Such a pair specifies a part of the lambda structure, namely all nodes which are below u, but not below v. We write

$$\begin{array}{rcl} \mathsf{b}^-(u/v) & := & \{w \mid u\triangleleft^* w, \text{ but not } v\triangleleft^* w\} \\ \mathsf{b}(u/v) & := & \mathsf{b}^-(u/v) \cup \{v\} \end{array}$$

Two segments are considered parallel iff a *correspondence function* between them exists and certain conditions on binding hold, which we cannot go into here due to lack of space.

Definition 1 *A* correspondence function *between two tree segments u/u' and v/v' is a bijective mapping $c : \mathsf{b}(u/u') \to \mathsf{b}(v/v')$ such that $u\triangleleft^* u'$, $v\triangleleft^* v'$, and for all nodes $w \in \mathsf{b}^-(u/u')$ and every label f of arity n it holds that:*

$$w{:}f(w_1,\ldots,w_n) \Leftrightarrow c(w){:}f(c(w_1),\ldots c(w_n)).$$

3 VP Ellipsis in CLLS

Now we briefly sketch how parallelism constraints are used to model ellipses. We consider the sentence (1), which is analyzed as the left-hand graph in Fig. 2. A lambda structure satisfying this constraint is shown on the right.

(1) Mary sleeps. John does too.

A correspondence function between the two segments can only exist if all nodes below X_1 have copies below Y_1. Except for X_2 (which is the hole), each copy must have the same label as the original. X_2 must correspond to Y_2, which has the label John.

Thus, parallelism constraints allow a very tight control over the copying process; different occurrences of Mary-labeled nodes are kept strictly apart, and no equivalent of a primary occurrence restriction is needed. As Egg et al. (2001) show, this analysis can deal with a large class of examples from the literature, including e.g. interactions with scope (Hirschbühler 1982) and antecedent-contained deletion. As for processing complexity, satisfiability of parallelism constraints is equivalent to context unification, whose decidability is unknown. Semi-decision procedures exist (Erk et al. 2001), and decidability of a linguistically relevant fragment is conjectured.

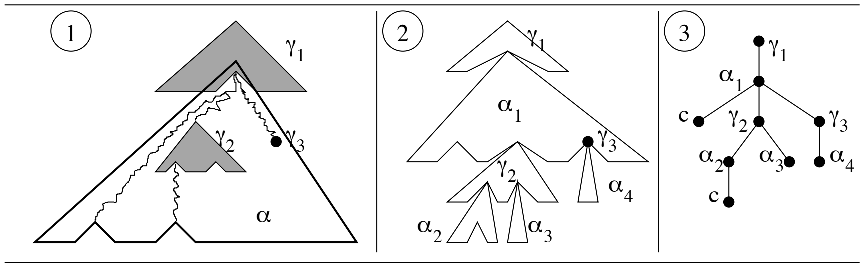


Figure 4: Jigsaw segments and alpha-gamma trees.

However, there are some examples which are a problem for CLLS.

(2) John went to the station, and every student did too, on a bike.

(3) Every man kissed his wife before John did.

The first sentence exhibits a scope ambiguity between "every student" and "a bike". The straightforward CLLS description excludes the weaker reading, as "on a bike" does not correspond to anything in the source sentence and hence, must take scope over the entire target clause. In the second sentence, the parallelism constraint forces "every man", the parallel element in the source clause, to be dominated by the root of the source clause. The reading where "every man" outscopes "before" is lost.

4 Jigsaw Parallelism

To represent these examples, we generalize ordinary CLLS parallelism to *jigsaw parallelism*, which allows much more fine-grained surgery of a lambda structure.

Jigsaw parallelism generalizes ordinary parallelism in two ways. First, segments α, γ are now allowed to have an arbitrary number (even 0) holes; we write $r(\alpha)$ for the root and $hs(\alpha)$ for the set of holes of α. $\mathsf{b}(\alpha)$ and $\mathsf{b}^-(\alpha)$ generalize straightforwardly; we also write $\mathsf{i}(\alpha)$ for $\mathsf{b}^-(\alpha) - \{r(\alpha)\}$. We call α a *singleton* iff $|\mathsf{b}(\alpha)| = 1$ and say that segments α, β *overlap properly* iff $\mathsf{b}^-(\alpha) \cap \mathsf{b}^-(\beta) \neq \emptyset$.

More interestingly, jigsaw parallelism allows us to exempt parts of segments from the parallelism condition; we "cut out" smaller segments from the larger, parallel one. In the simplest case (with a single removed segment), the result is as defined below.

Definition 2 *Let α, γ be segments of the same lambda structure. Then $\alpha - \gamma$ is a set of segments defined as follows.*

1. $\alpha - \gamma = \{\}$ *if* $\mathsf{b}(\alpha) \subseteq \mathsf{b}(\gamma)$.
2. $\alpha - \gamma = \{\alpha\}$ *if either α and γ do not overlap properly, or α is a singleton with* $\mathsf{b}(\alpha) \not\subseteq \mathsf{b}(\gamma)$.
3. *For non-singleton α to which the first two cases do not apply, let*

$$\mathsf{ro}(\alpha - \gamma) = \big(\{r(\alpha)\} - \mathsf{b}^-(\gamma)\big) \cup \big(hs(\gamma) \cap \mathsf{i}(\alpha)\big)$$
$$\mathsf{ho}(\alpha - \gamma) = \big(hs(\alpha) - \mathsf{i}(\gamma)\big) \cup \big(\{r(\gamma)\} \cap \mathsf{i}(\alpha)\big)$$
$$\mathsf{ch}(u, \alpha - \gamma) = \{v \in \mathsf{ho}(\alpha - \gamma) \mid u \triangleleft^+ v \textit{ and } \nexists u' \in \mathsf{ro}(\alpha - \gamma) \textit{ such that } (u \triangleleft^+ u' \triangleleft^+ v)\}$$

Then

$$\alpha - \gamma = \{u_0/u_1, \ldots, u_n \mid u_0 \in \mathsf{ro}(\alpha - \gamma), \{u_1, \ldots, u_n\} = \mathsf{ch}(u_0, \alpha - \gamma) \textit{ ordered left to right}\}.$$

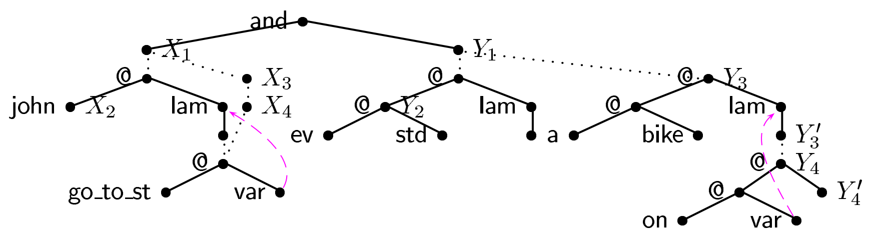
In the definition, α is broken into pieces by cutting out γ (hence the name). No two members of $\alpha - \gamma$ overlap properly; all are contained within α, and together with γ, they cover all of α.

The definition can be extended to cut out multiple segments from α. Let $\alpha_1, \ldots, \alpha_n, \gamma$ be segments of the same λ-structure such that for all $1 \leq i < j \leq n$, α_i and α_j do not properly overlap. Then $\{\alpha_1, \ldots, \alpha_n\} - \gamma := \bigcup_{i=1}^n \alpha_i - \gamma$.

Definition 3 *Let $\alpha, \gamma_1, \ldots, \gamma_n$ be segments of the same λ-structure. Then $\omega = \alpha - (\gamma_1, \ldots, \gamma_n) := \Big(\big((\alpha - \gamma_1) - \gamma_2\big) \ldots\Big) - \gamma_n$ is a* jigsaw segment.

We call the elements of ω *alpha segments* for short, and we call the excluded segments *gamma segments*. Also, we write $\mathsf{b}(\omega) = \bigcup_{\alpha' \in \omega} \mathsf{b}(\alpha')$. The above observations on overlap and coverage still hold. Note that the order in which gamma segments are subtracted does not matter.

The process of cutting out gamma segments is illustrated in Fig. 4. (1) is a schematic diagram of a segment α with two holes, from which segments $\gamma_1, \gamma_2, \gamma_3$ are being cut out. γ_1 overlaps only partially with α, and γ_3 is a singleton segment. If we compute the set $\{\alpha_1, \alpha_2, \alpha_3, \alpha_4\} = \alpha - (\gamma_1, \gamma_2, \gamma_3)$ according to Def. 3, we obtain a picture as in (2): α is cut along the gammas. Note that adjacent segments generally share a node, which is the root of one and a hole of the other segment.



$$X_1/X_2 - (X_3/X_3, X_4/X_4) \sim Y_1/Y_2 - (Y_3/Y_3', Y_4/Y_4')$$

Figure 5: Constraint for sentence (2).

The way that the alpha and gamma segments are plugged into each other is represented in the *alpha-gamma tree* shown in (3). An alpha-gamma tree is a tree which contains exactly one node with label α_i for each i, at most one node with label γ_i for each i, and nodes with label $\bullet$ for the holes of α. The children of each node are the segments plugged into the holes of the corresponding segment, in the correct left-to-right order. It can be shown that if the gamma segments do not overlap properly, such an alpha-gamma tree always exists. If they exist, alpha-gamma trees are unique up to permutations of equal singleton gamma segments.

Now we can use alpha-gamma trees to define *jigsaw correspondence functions*, correspondence functions between jigsaw segments. Two jigsaw segments correspond iff first, gamma segments with the same index are in the same positions in the alpha-gamma trees, and second, alpha segments in the same positions correspond in the ordinary sense. (Alpha-gamma trees are useful precisely because they provide an exact way of saying that segments are "in the same positions".)

Definition 4 *A* jigsaw correspondence function *between jigsaw segments ω and ω' is a bijective mapping $c : \mathsf{b}(\omega) \to \mathsf{b}(\omega')$ which satisfies the following conditions:*

1. *There are isomorphic alpha-gamma trees θ, θ' of ω and ω'; call the (tree) isomorphism h.*
2. *Every node u of θ is labeled γ_j iff $h(u)$ is labeled γ_j'; u is an alpha node iff $h(u)$ is; and u is labeled $\bullet$ iff $h(u)$ is.*
3. *For every i, the restriction of c to $\mathsf{b}(\alpha_i)$ is an ordinary correspondence function between α_i and $h(\alpha_i)$.*

Jigsaw parallelism is obtained by simply replacing the words "segment" and "correspondence function" in the definition of ordinary parallelism by "jigsaw segment" and "jigsaw correspondence function". We extend the syntax of CLLS by *jigsaw parallelism literals* in the straightforward way.

5 VP Ellipsis Using Jigsaw Parallelism

We will now apply jigsaw parallelism constraints to describe the meanings of (2) and (3).

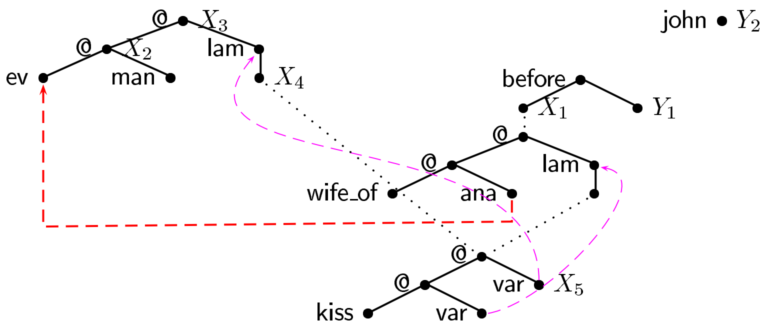
A constraint for (2) is shown in Fig. 5; X_1 will denote the root of the source sentence, Y_1 that of the target sentence. As in Fig. 2, the constraint contains an explicit description of the source clause, descriptions of the parallel elements in the target clause, and a (jigsaw) parallelism constraint. The first parts of this constraint (before the minus symbols) work exactly as before, establishing equality of the semantics except for "John" and "every student".

But now jigsaw parallelism allows us to exempt the two segments that constitute the meaning of "on a bike" from the parallelism. This is done by subtracting the segment terms Y_3/Y_3' and Y_4/Y_4' on the right-hand side. The corresponding gamma segments on the left-hand side are simply appropriate singletons.

The analysis of (3) is shown in Fig. 6. Ignoring the anaphoric reference here (which leads to a strict/sloppy ambiguity correctly resolved through the mechanisms in (Egg et al. 2001)), let us take a quick look at the reading in which "every man" takes narrowest scope and the reading in which "every man" takes widest scope (even over the "before").

In the narrowest-scope reading, X_2 is dominated by X_1, and the first jigsaw literal behaves exactly like the ordinary parallelism constraint $X_1/X_2{\sim}Y_1/Y_2$, which is the original analysis. The second literal is necessarily satisfied in this case.

The wide-scope reading, with $X_4 \lhd^* X_1$, is more interesting. Consider the first conjunct, which now enforces that the lambda structures below X_1 and X_2 must be completely parallel. As the denotation of $X_2/$ is not part of the alpha-gamma tree of the left-hand jigsaw segment, Y_2 cannot be below X_2 either. The second conjunct, involving fresh variables Y_3, Y_4, ensures that the application and abstraction around "every man" have correspondents in the target semantics; these correspondents must be around "John" because Y_2 is a hole of this segment.



$$X_1/-X_2/\sim Y_1/-Y_2/ \ \wedge\ X_3/X_2,X_4 \sim Y_3/Y_2,Y_4$$

Figure 6: Constraint for sentence (3)

The correct binding of the correspondent of X_5 is ensured through the binding rules we have omitted in the definition of parallelism. This binding also automatically entails that Y_4 really dominates this correspondent.[1]

6 Conclusion

We have shown how to represent some previously difficult cases of VP ellipsis in an extension of CLLS. We have achieved this by defining *jigsaw parallelism*, a generalization of the original parallelism in CLLS which allows to cut out parts of the corresponding segments.

Jigsaw parallelism is a very versatile tool for tree surgery, which we expect has uses outside of ellipsis. For instance, the TAG operations of substitution, adjunction, and sister adjunction can all be represented using jigsaw parallelism constraints. Adjunction, for example, is expressed by the constraint $\alpha - \gamma \sim \alpha' - \gamma'$ where α, α' are the complete trees before and after the adjunction, γ is a singleton segment at the adjunction site, and γ' is the tree that is to be adjoined.

The next question that needs to be considered is how to process jigsaw parallelism constraints. As soon as it is known whether the gamma segments are in α or not, jigsaw parallelism can be almost completely reduced to a *group parallelism constraint* (Bodirsky et al. 2001) of the alpha segments, but there are some subtleties with binding. This should make it easy to obtain a sound and complete solution procedure from known procedures from group parallelism, but we believe that jigsaw parallelism avoids some hard instances of group parallelism and may thus be amenable to more efficient specific techniques.

[1]In fact, we have been somewhat imprecise here, again for lack of space. The original definition of parallelism does not allow binding across different parallelism literals. This is remedied by using *group parallelism* (Bodirsky et al. 2001), into which jigsaw segments and correspondences can be plugged just as easily.

Finally, while we have shown how to *represent* some cases of ellipsis, we have said nothing about how to *obtain* these representations from a syntactic analysis. First steps towards this goal in the CLLS framework have been taken in (Egg and Erk 2001); we expect that the great flexibility of jigsaw parallelism will simplify the interface design.

References

Bodirsky, M., K. Erk, A. Koller, and J. Niehren (2001). Beta reduction constraints. In *Proceedings of RTA*, Utrecht, The Netherlands. To appear.

Crouch, R. (1995). Ellipsis and quantification: A substitutional approach. In *Proceedings of the 7th EACL*, Dublin, 229–236.

Dalrymple, M., S. Shieber, and F. Pereira (1991). Ellipsis and higher-order unification. *Linguistics & Philosophy 14*, 399–452.

Egg, M. and K. Erk (2001). A compositional account of VP ellipsis. Submitted, available at `http://www.ps.uni-sb.de/Papers/abstracts/ellhpsg01.html`.

Egg, M., A. Koller, and J. Niehren (2001). The constraint language for lambda structures. *Journal of Logic, Language, and Computation 10*. To appear, available at `http://www.ps.uni-sb.de/Papers/abstracts/clls2000.html`.

Erk, K. and A. Koller (2001). VP ellipsis by tree surgery. Extended version at `http://www.ps.uni-sb.de/Papers/abstracts/jparal.html`.

Erk, K., A. Koller, and J. Niehren (2001). Processing underspecified semantic representations in the constraint language for lambda structures. *Journal of Language and Computation*. To appear.

Gardent, C. and M. Kohlhase (1996). Higher-order coloured unification and natural language semantics. In *Proceedings ACL'96*.

Hirschbühler, P. (1982). VP deletion and across the board quantifier scope. In J. Pustejovsky and P. Sells (eds), *NELS 12*, Univ. of Massachusetts.

Koller, A., J. Niehren, and K. Striegnitz (2000). Relaxing underspecified semantic representations for reinterpretation. *Grammars 3*(2–3). Special issue on MOL 6.