LOGON

Technical Report # 2007-11

The Transfer Formalism

General-Purpose MRS Rewriting

(Draft of November 30, 2008)

Stephan Oepen

Department of Informatics

University of Oslo

Contents

1 Background — Desiderata

Two of the core assumptions in transfer-based MT are that (i) translational equivalences are expressed with reference to abstract, structured representations, and that (ii) the source and target language structures for translationally equivalent expressions need not be identical, nor in the general case even be isomorphic to each other. With its design decision for transfer at the representational level of semantics, and specifically the Minimal Recursion Semantics (MRS) framework, the objects manipulated in the transfer component are semantic structures (MRSs). Given an input MRS (specific to the source language, henceforth SL), transfer constructs one or more translationally equivalent MRSs—each, in turn, specific to the target language (TL). Quite generally speaking, a transfer grammar thus establishes a relation between source and target language semantic structures.

A large part of transfer, actually, is mere re-labeling, replacing symbols in the SL-specific semantics with equivalent[1] TL symbols. However, as discussed in Chapter **??**, such re-labeling can constitute a one-to-many relation, and it may need to be conditioned on a specific context of use. Even when just replacing one predicate symbol with another one—say the Norwegian _utover_p (a prepositional relation) with either _across_p or _during_p in English, depending on whether the internal argument to the preposition is of a temporal kind—it can be necessary to attach contextual conditions on statements about transfer correspondences.

Chapter **??** already suggested examples of translational equivalences that involve linguistic structures that differ between the source and target languages, in some cases only mildly so, in others quite fundamentally. Such examples are commonly dubbed *translational divergences*. Besides the potential need for one-to-many and conditional transfer, translational divergences provide the main motivation for approaches to transfer that employ formally quite powerful machinery, viz. realizing the transfer relation as a *resource-sensitive structural rewrite system*. In the case of LOGON, the structures manipulated by transfer are MRSs, which abstractly constitute directed acyclic graphs.[2] Thus, the LOGON transfer formalism

[1] For the remainder of this chapter, we will often use just the term *equivalence* when referring to relations of *translational* equivalence, i.e. the idealized relation of perfect translation. In much the same spirit, we will use the term (transfer) *correspondence* to refer to the equivalence relation specifically at the level of transfer, i.e. as the idealized relation represented by the transfer grammar.

[2] At this level of abstraction, MRSs actually have much in common with feature structures, as used for example in the LOGON parsing and generation grammars. However, it is important to remember that there are important differences between MRSs and the various brands of feature structures used in LFG or HPSG research. Among these are the lack of recursion (one MRS can never embed another MRS), the very limited use of typing, specialized notions of underspecification, and the central reliance on unordered bags (i.e. multi-sets), which most feature logics do not provide as a descriptive means.

provides a special-purpose graph rewriting system, where individual rewrite rules relate (parts of) a source language MRS to the corresponding (parts of a) target language MRS. In a manner of speaking, thus, transfer rewrites the SL MRS (into one or more TL MRSs) in a stepwise, piecemeal fashion—each step processing bite-sized parts (and sometimes relatively large chunks) of structure. The rewrite system is *structural* quite simply in the sense that input and output specifications alike can exploit arbitrary structural properties of the MRS graphs. The system is *resource-sensitive* in the sense that rules 'consume' their input, replacing it with the corresponding target language structure. This contrasts with, for example, standard parsing methods for phrase structure grammar frameworks, where rules combine constituents to form larger units but never consume their inputs, i.e. the component parts of larger constituents remain available for combination using other rules.

Finally, the LOGON transfer formalism faciliates what is often called 'feeding and bleeding' among rules. In a nutshell, there is no formal separation between SL and TL MRSs, in fact for most of the transfer process the system manipulates mixed structures: MRSs *still* containing some SL parts and simultaneously *already* containing some TL parts. Thus, the output of one rule will be accessible as input to other rules; in principle the formalism makes it possible for one rule to insert material, which another rule may then delete again, at a later point in the rewrite process. Such 'intermediate' (or temporary) information makes it possible to provide contextual (or status) information to groups of transfer rules. As we will discuss in considerable detail in Section 5 below, the resource-sensitive approach to transfer entails that the ordering of transfer rules matters: after all, invocation of each rule changes the 'state of the universe' (i.e. the current MRS) visible to subsequent rule applications. In this respect, resource-sensitive rewriting is not a purely declarative framework (reverting the transfer relation is not straightforward and, in the general case, may not be possible).

We will defer an in-depth argumentation for this general approach (and comparison to alternate views) to transfer to Section 9 below, but it is fair to say that a majority of transfer-based MT systems make use of structural rewriting; and a fairly large number of these, in turn, assume resource-sensitive rewriting. In terms of formal computational complexity, rewrite systems are equivalent to the most powerful general device of computation: the LOGON transfer component is a Turing machine, indeed. Accordingly, transfer in this paradigm has unattractive worst case complexity properties, but in Section 7 we argue that, nevertheless, it is possible to provide an efficient implementation of the formalism, at least for certain classes of transfer grammars. Unlike in grammar-based parsing and generation, there is less established common ground in terms of (semantic) transfer formalisms and algorithms. Therefore, this chapter has three main goals, viz. to (i) provide sufficient formal and practical background for the in-depth, linguistic discussion of

the LOGON transfer grammars in Chapter **??** and Chapter **??**; to (ii) serve as reference documentation for others to deploy the LOGON MRS rewriting system (be it for transfer or other tasks); and to (iii) at least summarize the key points of the underlying implementation, so as to allow the comparison to other such systems or to related techniques in, say, parsing and generation. The chapter makes comparatively heavy use of footnotes, aiming to background the discussion of formal and technical details, so as to not disrupt the flow of the high-level presentation of the LOGON transfer formalism.

2 MRS Rewriting: The Basics

A transfer grammar is composed of a sequence of MRS rewrite rules, where we will henceforth often use the abbreviation MTR to refer to a single such rule. The general form of one MTR is shown somewhat schematically in (1), as a four-tuple with components $\mathcal{C}$ (context), $\mathcal{I}$ (input), $\mathcal{F}$ (filter), and $\mathcal{O}$ (output).

(1) $[\,\mathcal{C}\,:\,]\;\mathcal{I}\;[\,!\,\mathcal{F}\,]\rightarrow\mathcal{O}$

Here, the names of the four components (in italics) serve as placeholders: in actual transfer rules, each such component is a (typically partial) description of an MRS.[3]

The square brackets in (1) indicate optionality of select components, while the remaining symbols form a literal part of our MTR notion and serve to visually segregate the parts, viz. the context colon (':'), filter exclamation point ('!'), and output arrow ('→'). Optional MTR components will often be omitted in the presentation of example rules, unless they play a relevant role, of course.

Of the four components in transfer rules, $\mathcal{I}$ and $\mathcal{O}$ are most typical ones. For a rule R to be applied, the MRS description that constitutes the $\mathcal{I}$ component of R has to be compatible with the *current* MRS M_i, which can be either the original input given to transfer or the intermediate structure obtained from earlier MTR applications. Section 3 below spells out the formal details of matching MTR components to MRSs, but abstractly we can think of this process as an instance of graph unification: finding (all possible) ways of aligning two graph-structured objects—one a partial MRS description (the $\mathcal{I}$ component of R), the other an actual, complete MRS (M_i). In case $\mathcal{I}$ unification succeeds, we say that rule R 'fires', i.e. is invoked

[3]Throughout this chapter we will often blur the distinction between MRS *descriptions* and actual MRS *objects*. We use the same notation for both universes, where the transfer rule components constitute descriptions primarily in the sense that they admit a few additional descriptive devices—similar to pattern matching operators—that are not part of MRSs proper. The descriptions (or patterns) used in transfer rules typically are partial specifications, aiming to match only the sub-structure in the target (or input) MRS that is relevant to the rule in question.

successfully on M_i. All sub-structure of M_i that was aligned with the $\mathcal{I}$ component of R is consumed, and the $\mathcal{O}$ specification determines what to insert in its place. Effectively, once the application of R is complete, $\mathcal{O}$ will have replaced $\mathcal{I}$, and the result is a new intermediate structure M_{i+1}. When assembling M_{i+1} as the result of a successful rule application, the $\mathcal{O}$ component has access to the structural alignments created during unification of the $\mathcal{I}$ description to M_i. Therefore, rules can effectively establish bindings to $\mathcal{I}$ sub-structures during unification and 'carry over' matching parts of M_i into M_{i+1}, for example replacing the predicate symbol of an elementary predication (EP) but preserving the logical variables of arguments in that EP. For the construction of M_{i+1}, finally, all remaining parts of M_i—sub-structures not aligned with any of the $\mathcal{I}$ description—are preserved as is, that is copied over without changes.

Consider (2) as an example of about the most basic MTR flavor.

(2) $\langle \_, \{ \boxed{h_0}\!:\!\texttt{\_bekk\_n}(\boxed{x_0}) \}, \{ \} \rangle$
$\rightarrow \langle \_, \{ \boxed{h_0}\!:\!\texttt{\_creek\_n\_1}(\boxed{x_0}) \}, \{ \} \rangle$

This rule is a simple replacement of the Norwegian predicate _bekk_n with the English _creek_n_1, assuming that both correspond to plain nominal semantics (i.e. are simple one-place relations). Our example rule (2) has $\mathcal{I}$ and $\mathcal{O}$ components, but no $\mathcal{C}$ or $\mathcal{F}$ conditions. See Chapter **??** for more details on the MRS framework and notation, but note that the 'boxed' elements here constitute a new type of variable, viz. transfer-level *meta-variables*. Intuitively, the operation of rule (2) is really straightforward: its $\mathcal{I}$ component will look for one elementary predication with predicate _bekk_n, a handle, and one argument ($\boxed{h_0}$ and $\boxed{x_0}$, respectively). The $\mathcal{O}$ specification replaces the predicate but preserves the handle and argument: whatever the values were prior to invocation of the rule, they will remain unchanged once the predicate symbol has been replaced. Furthermore, except for the one EP matched by the rule, everything else in the current MRS will be left unchanged, specifically the top handle, other elementary predications, and handle constraints. In (2), we use the notation '_' and { } to indicate the *lack* of constraints on the top handle and bag of handle constraints, respectively.

Note that MRS variables are typed, but the particular labels we use for MTR meta-variables are formally unrelated to such type constraints on actual (i.e. object-level) MRS variables; these labels are mere identifiers, where for increased readability transfer rules often opt for labels that suggest the intended use. Therefore, the tag $\boxed{x_0}$ can actually unify with any kind of MRS object variable, i.e. rule (2) could hypothetically also apply to usages of _bekk_n where its argument were an event (instead of the referential index that we typically associate with nominals). To make the rule more specific in this particular respect, one could rewrite its $\mathcal{I}$ component as $\boxed{h_0}h\!:\!\texttt{\_bekk\_n}(\boxed{x_0}x)$, thus including explicit type constraints on object

variables. In practice, it is an interesting methodological issue how to determine the adequate degree of specificity in transfer rules. In our current example, the MRS framework guarantees that EP handles will always be of variable type h, so putting an additional type constraint on $\boxed{h_0}$ is superfluous.[4] Likewise, albeit not formally required, it would be a potentially troubling state of affairs if predicates associated exclusively with nominals were to use events as their inherent argument; in this case, the transfer grammar might decide to assume such basic wellformedness of input MRSs and not put redundant or unnecessary constraints into the rule system.

With $\mathcal{I}$ and $\mathcal{O}$ as the core components of each MTR, the optional $\mathcal{C}$ provides a way of conditioning a rule on specific sub-structures in the current MRS—much like $\mathcal{I}$, but without consuming what is matched during unification of the $\mathcal{C}$ component. Conversely, $\mathcal{F}$ can be viewed as a negative context condition: whenever unification of the $\mathcal{F}$ component succeeds, application of the particular rule will be blocked. Just as $\mathcal{I}$, both $\mathcal{C}$ and $\mathcal{F}$ are partial descriptions of MRSs, and structural alignments established during unification of $\mathcal{C}$ parts are accessible for $\mathcal{O}$ assembly—by means of transfer meta-variables.

Example (3) demonstrates the use of $\mathcal{C}$ conditions in a slightly more complex transfer rule.

(3) $\langle \_, \{ \text{\_tur\_n}(\boxed{x_1}) \}, \{ \} \rangle : \langle \_, \{ \boxed{h_0}\text{:\_gå\_v}(\boxed{e_0}, \boxed{x_0}, \boxed{x_1}) \}, \{ \} \rangle$
$\rightarrow \langle \_, \{ \boxed{h_0}\text{:\_take\_v\_1}(\boxed{e_0}, \boxed{x_0}, \boxed{x_1}) \}, \{ \} \rangle$

This rule accounts for the fact that the predicate associated to the Norwegian verb *gå* (to 'go' or 'walk' in English) translates as the light verb 'take' in case its ARG2 argument—the one corresponding to the direct object in this case[5]—is introduced by the predicate _tur_n, which would correspond to the nouns 'walk' or 'trip' in English. Putting the constraint that determines the choice of $\mathcal{O}$ predicate into the $\mathcal{C}$ component effectively conditions the rule on the combined structural configuration required in its left-hand side (where, again, meta-variables establish bindings across MTR components). A successful application of rule (3), however, will preserve the conditioning _tur_n EP in the current MRS, such that it can be transferred separately—using the same rule or rules that account for all other occurrences of that predicate. Finally, observe that conditioning at the level of semantics makes

[4] In fact, the use of typing and inheritance in the LOGON specification language for transfer rules provides an easy means of enforcing constraints of this type globally; see Section 4 below.

[5] Remember that MRS role labels start from ARG0, where most relations associated to open-class words use ARG0 for their inherent argument, e.g. an event ($\boxed{e_0}$) in the case of verbs and (predicative) adjectives, and a referential index ($\boxed{x_1}$) in the case of nominals. For increased intellectual stimulation, we will at times refer to either kind of inherent argument (ARG0 values) as just the *index*, thus generalizing over the event vs. referential index distinction. We further say that a variable is *introduced* by a predicate when it serves as the index in an elementary predication with that predicate symbol.

our rule (3) independent of syntactic (or other surface) parameters of variation, whether the actual argument to _gå_v were itself a compound (*søndagstur*, say: 'sunday walk' or 'summer trip'), for example, or whether it might be positionally separated from the verb by virtue of being topicalized.

Finally, rule (4) shows an example of multiple EPs, combined with a new handle constraint, in the right-hand side of transfer rules, i.e. the $\mathcal{O}$ description.

(4) $\langle \_, \{ \boxed{h_0}\text{:\_seter\_n}(\boxed{x_0}) \}, \{ \} \rangle$
$\rightarrow \langle \_, \left| \begin{array}{l} \_\text{:implicit\_q}(x_1, h_1, \_), \\ h_2\text{:\_mountain\_n\_1}(x_1\{\text{NUM } sg\}), \\ \boxed{h_0}\text{:\_pasture\_n\_1}(\boxed{x_0}), \boxed{h_0}\text{:unspec}(\_, \boxed{x_0}, x_1) \end{array} \right| \{ h_1 =_q h_2 \} \rangle$

This rule translates (the semantics associated with) the Norwegian *seter* as 'mountain pasture' in English, i.e. it presents an instance of (mild) translational divergence: source language lexicalization corresponds to target language decomposition—in this case as a nominal compound.

The MRS account of compounds straightforwardly rests on a vague two-place unspec relation (as in most cases the grammar has little to say about the specific relation between the two parts) holding between the two parts of the compound, much like an intersective modifier on the head. The semantic head in this example is _pasture_n_1, and it assumes the handle and index (ARG0 value) of the sole $\mathcal{I}$ element. In addition to the compound head and the vague two-place relation, the rule further introduces the compound modifier _mountain_n_1—as the second argument of the unspec relation—and adds an underspecified quantifier binding the new referential index x_1 (see Chapter **??** for the rationale behind this additional quantifier). Finally, note that the $\mathcal{O}$ component of rule (4) creates several new variables (h_1, h_2, and x_1), and (albeit in part for expository reasons) it carefully determines the MRS variable types on each of them, and further constrains x_1 to bear singular number.

3 MRS Subsumption and Unification

To understand how exactly the left-hand side of a transfer rule is matched against the current MRS, we need to introduce the concepts of MRS subsumption and unification first. These are, in turn, closely related to various kinds of underspecification in the MRS framework. Copestake, Flickinger, Pollard, & Sag (2005) discuss the MRS account of scope underspecification (by means of handle constraints); conversely, our emphasis in the discussion of underspecification will be on other aspects of partiality in MRSs, which to our best knowledge have not been discussed previously. We will inductively develop a semi-formal notion of MRS subsumption,

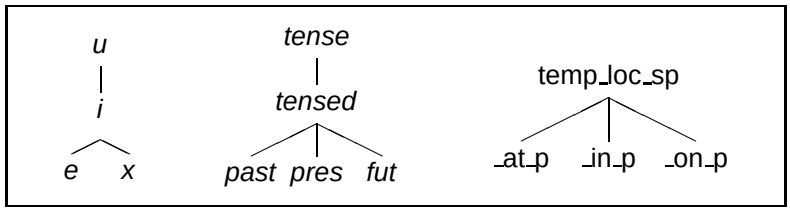

Figure 1: Hierarchical relations among (i) MRS variable types, (ii) select variable properties, and (iii) a small cluster of related predicate symbols.

which then serves to define the operation of unification. Intuitively, we will adapt core concepts from general graph theory to the MRS universe: a graph G_1 is said to subsume another structure G_2 if and only if a mapping can be provided from nodes in G_1 to (corresponding) nodes in G_2, such that all edges among nodes of G_1 also exist (with compatible labels) among the mapped nodes in G_2.[6].

The basic building blocks of MRSs are logical variables, which optionally can include a (flat) set of variable properties, ordered pairs of property labels and values. Figure 1 shows an excerpt from the hierarchies of (i) MRS variable types, (ii) variable properties, and (iii) predicate symbols. Such hierarchical organization facilitates underspecification of the corresponding MRS elements. We can now say, for example, that *i* subsumes both *e* and *x*, or that *e*{TENSE *tensed*} subsumes *e*{TENSE *past*}. Conversely, there is no subsumption between, say, *e*{TENSE *past*} and *e*{TENSE *pres*}.

Moving on to the level of individual EPs, h_0:_temp_loc_sp(e_0, e_1, x_0) subsumes h_0:_at_p(e_0, e_1, x_0), and so does _:_at_p(_, e_1, _). In the latter example, we use the transfer-level wildcard operator ('_', an anonymous meta-variable) to indicate uninstantiated arguments. As MRS transfer supports 'variable-arity' relations (in a technical sense), omitting such arguments (at least where order coding allows it without ambiguity) would be equivalent, e.g. h_0:_at_p(_, e_1), or—using explicit role labels—h_0:_at_p(ARG1 e_1). In summary, MRS variables and EPs behave much like (non-recursive, untyped) feature structures as regards the subsumption relation.

Examples (5) to (8) present a series of full, multi-EP MRSs. Let us ignore scopal relations and handle constraints for the time being. By extension of EP subsumption, (5) straightforwardly subsumes (7).[7]

[6]Due to space constraints we do not elaborate on which MRS elements should correspond to nodes or edges in the graph analogy. In fact, multiple such views are possible, often with equivalent properties regarding the subsumption relation; see Fuchss, Koller, Niehren, & Thater (2004) for a candidate account.

[7]Remember that EPs form an un-ordered multi-set, such that (7) could be rendered equivalently as: $\langle h_0, \{ h_0$:_snow_n_1(x_0) h_0:_fresh_a_1(e_0, x_0) $\}, \{\} \rangle$.

The same holds between (6) and (7), though in a slightly more interesting configuration: here, the argument in the _fresh_a_1 relation actually is an instantiated variable (x_0), distinct from the corresponding variable in (7). However, specializing variables (when establishing a mapping of nodes between sub-graphs) can extend beyond refining variable types; we can also equate a variable with another one (assuming the types and variable properties are compatible). This latter step corresponds to creating a new reentrancy in the graph, again in close analogy to the universe of feature structures.

(5) $\langle h_0, \{ h_0{:}\text{\_fresh\_a\_1}(\_, x_0), h_0{:}\text{\_snow\_n\_1}(x_0) \}, \{ \} \rangle$

(6) $\langle h_0, \{ h_0{:}\text{\_fresh\_a\_1}(e_0, x_0), h_0{:}\text{\_snow\_n\_1}(x_1) \}, \{ \} \rangle$

(7) $\langle h_0, \{ h_0{:}\text{\_fresh\_a\_1}(e_0, x_0), h_0{:}\text{\_snow\_n\_1}(x_0) \}, \{ \} \rangle$

(8) $\langle h_0, \{ h_0{:}\text{\_snow\_n\_1}(x_0) \}, \{ \} \rangle$

It might seem intuitive to say that MRS (8) also subsumes (7): in terms of 'informational content', (7) clearly contains all the information of (8), plus additional information provided by the extra EP. When matching the left-hand side of a transfer rule against the current MRS M_i, however, this notion of partiality is already built into the rewrite formalism. For a rule to fire, it is sufficient for it to match *part* of M_i, where extra EPs in M_i not aligned with any part of the rule will not be effected by the application of the rule. Therefore, our notion of MRS subsumption is defined more narrowly: for two structures to stand in the subsumption relation, they need to have the same number of EPs. While still ignoring scope underspecification, this design decision means that underspecification either applies 'within' one relation, or in the way EPs are connected to each other (and the top handle).

Unfortunately, the graph containment analogy for MRS subsumption breaks down once we take into account handle constraints; in principle, there can be multiple distinct ways of scope underspecification in an MRS, all describing (*link-subsuming* in the terminology of Copestake et al., 2005) the same set of scope-resolved structures. This is in part because of the interactions of handle constraints and other, implicit wellformedness conditions on MRSs, in part because the general framework allows additional kinds of handle constraints besides the 'equality modulo quantifiers' relation ('$=_q$') discussed so far. In the general case, comparing two MRSs for subsumption would require one to fully resolve scope underspecification, i.e. 'unfold' each underspecified structure into the corresponding set of scope-resolved MRSs (Copestake et al., 2005).[8] In LOGON, transfer (and generation)

[8]The LOGON grammars, in fact, restrict themselves to only $=_q$ handle constraints. Thus, if one were to formally disallow other kinds of handle constraints, there would be a bounded maximum number of possible handle constraints for a given structure, and the subsumption relation would be computable, in principle, without enumerating all scope-resolved structures.

operate on semantic structures that leave scope ambiguities unresolved—after all, this is one of the main attractions of MRS-like approaches to semantics for these applications. We thus make a limiting assumption regarding handle constraints, viz. that (i) for two MRSs to stand in the subsumption relation, they need to have an equal number of handle constraints, and that (ii) handle constraints are restricted to just the $=_q$ kind. Combined with additional constraints imposed by the MRS framework—that handle constraints always relate a handle in argument position to one labeling an EP—these assumptions mean that, for the purpose of testing MRS subsumption in transfer, there cannot be partiality of handle constraints.[9] Most grammars with a semantics component based on MRS make use of $=_q$ constraints in a strongly codified manner, hence in practice divergences at this level are rare (see Chapter **??** for background).

With a notion of MRS subsumption at hand, we can now define the *unification* of two MRSs, M_1 and M_2, as the most general structure M_3 that is subsumed by both M_1 and M_2. Unification fails when no such structure exists. Intuitively speaking, unification of two MRSs preserves all information from the input structures and does not add any new information. Reusing our earlier examples, MRSs (5) and (6) have (7) as their unification. Conversely, (5) and (8) do not unify. By default, matching the left-hand side of a transfer rule R_i to the current MRS M_j computes the *joint* unification of the $\mathcal{C}$ and $\mathcal{I}$ components in R_i with M_j. Procedurally, this actually is a sequence of two unifications, but conceptually the totality of $\mathcal{C}$ and $\mathcal{I}$ is treated as a single graph, such that one consistent set of bindings (mapping of graph nodes) is established.

It follows from how we defined subsumption that each EP in M_j must unify against exactly one EP in $\mathcal{C}$ or $\mathcal{I}$, i.e. although we allow unification to equate distinct (meta-)variables, EP-level correspondences must be one-to-one (in Section 6 below, we will see other ways in which EPs are treated as units with a somewhat special status). Given the negative polarity of the $\mathcal{F}$ component, it is relevant do say whether $\mathcal{F}$ is tried *before* or *after* $\mathcal{C}$ and $\mathcal{I}$ matching. Parts of M_j compatible with $\mathcal{F}$ might also match $\mathcal{C}$ or $\mathcal{I}$, such that the rule blocking effect of a $\mathcal{F}$ specification can depend on when it is evaluated. In practice, LOGON transfer grammars make very limited use of $\mathcal{F}$ conditions, and it was found most useful to test these once all other left-hand side components of a transfer rule have been matched, i.e. after both $\mathcal{C}$ and $\mathcal{I}$.

[9] See Chapter **??** for a discussion of related concepts in generation. The original test for MRS equivalence (i.e. mutual subsumption) in the LKB implementation actually ignored handle constraints completely.

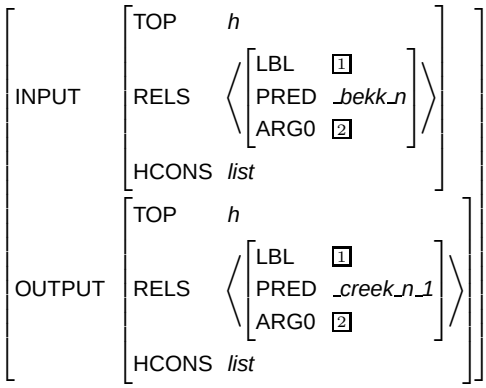

Figure 2: Feature structure representation of a simple transfer rule.

4 Typing in Transfer Grammars

Our first MTR example (2) above may seem a surprisingly complex statement, seeing that it records the simple fact that the Norwegian noun *bekk* translates as 'creek' in English. First, recall that transfer in LOGON operates at the level of semantics; in this and other chapters, we will typically discuss relations of translational correspondence in terms of semantic predicates (and their arguments), rather than in terms of words.[10] Second, and more importantly, there are large numbers of transfer correspondences like the one exemplified in (2), each replacing the predicate symbol while preserving the handle and inherent argument of one elementary predication. For descriptive adequacy (and convenience), it is desirable to separate the information about the general *pattern of correspondence* from the *actual instances* of that pattern.

For the purpose of stating *generalizations* over transfer rules, the LOGON approach builds on the successful experience in the Head-Driven Phrase Structure Grammar (HPSG) framework, and specifically the methodology that has enabled the creation of large HPSG implementations like the ERG (see Chapter **??** for background). Abstractly similar to common techniques in object-oriented software de-

[10]The vast majority of predicates in the semantics, of course, are associated to lexical units (words or multi-word expressions), and the few predicates associated to grammatical processes (e.g. derivation or causativization) and constructions (e.g. compounding or coordination) tend to contribute more abstract relations. However, many syntactic distinctions are either bleached at the level of semantics or only reflected indirectly. Adjectives and adverbs—'quick' and 'quickly', for example—and distinct verb frames related by diathesis alternations or optionality of complements often share a common semantic predicate. As pointed out in Chapter **??** already, we compose open-class predicate names with suffixes like _n or _v, but this is merely a notational convention to obtain a consistent top-level sense division.

velopment, our main tool in developing transfer grammars is the use of (typed) feature structures (TFSs) and multiple inheritance. Chapter **??** introduced the correspondence between the TFS and MRS universes. In a nutshell, Figure 2 presents an alternate way of encoding the information of our first transfer rule (example (2) above).[11]

We will discuss the TFS to MRS mapping in more detail shortly, but note how the top-level features correspond to the individual components of MTRs (in this case, $\mathcal{I}$ and $\mathcal{O}$). Comparing rule (2) and the feature structure of Figure 2—at least if judging by the amount of ink on the page—we have hardly increased the readability of our example. The correspondence to the TFS universe, however, is relevant in that it allows us to 'piggyback' on the general logic of typed feature structures, whose mathematical and computational properties are thoroughly studied (Rounds & Kasper, 1986; Carpenter, 1992; inter alios).

Much in the way grammars like the ERG organize linguistic knowledge, we can thus state the generalization over transfer rules like (2)—replacing the predicate in plain one-place relations—as an abstract type. For the present example, a type *noun_mtr*[12] can be defined to correspond to the full feature structure of Figure 2, with the exception of the specific PRED values, i.e. the predicate symbols proper. Assuming this type, we can reduce the actual transfer rule to a statement like (9), where the conjunction operator ('∧') simply means to combine (i.e. inherit) all constraints from the supertype *noun_mtr* with the additional, 'local' TFS specification.

(9) *noun_mtr* ∧

$$\begin{bmatrix} \text{INPUT}|\text{RELS} & \left\langle \left[\text{PRED} \;\; \textit{\_bekk\_n} \right] \right\rangle \\ \text{OUTPUT}|\text{RELS} & \left\langle \left[\text{PRED} \;\; \textit{\_creek\_n\_1} \right] \right\rangle \end{bmatrix}$$

Example (9) is relatively close to how transfer rules are actually encoded in the LOGON transfer grammars. Going back to the slightly more complex example (4) above, (10) shows the actual source code of this rule—presupposing the existence of an abstract correspondence type *n_n+n_mtr* for one (lexicalized) nominal rela-

[11] Although we make central use of types in setting up hierarchical (i.e. inheritance) relations among relevant semantic configurations and parts of transfer rules, the actual types at the leaf of the hierarchy have no important bearing on the correspondence to MRS elements. Hence we will at times suppress type information (or features with irrelevant values) in the presentation of feature structure examples.

[12] The suffix *_mtr* (short for 'MRS transfer rule') in LOGON transfer grammars indicates the yield of the type hierarchy, i.e. those correspondence types expected to be instantiated by actual transfer rules. Note that the information comprised in the type *noun_mtr* may of course be further decomposed by means of additional, more general, supertypes or even the use of multiple inheritance.

tion translating as a nominal compound. Note that the TFS descriptions of MRSs encode the (logically) unordered multi-sets of elementary predications and handle constraints (RELS and HCONS in the TFS universe) as ordered lists, which makes it possible to 'overlay' information on specific elements by positional references.[13]

(10)
```
seter_mountain+pasture_n := n_n+n_mtr &
[ INPUT.RELS < [ PRED "_seter_n_rel" ] >,
  OUTPUT.RELS < [ PRED "_mountain_n_1_rel" ],
                [ PRED "_pasture_n_1_rel" ], ... > ].
```

The LOGON transfer component is built as an extension of the Linguistic Knowledge Builder (LKB; Copestake, 2002; see Chapter **??** for background). Accordingly, the specific syntax of (9) is that of the Type Definition Language ($\mathcal{TDL}$; Krieger & Schäfer, 1994), but by comparison to examples (9) and (2) above it should be reasonably clear how to interpret this definition.[14] Comparing (10) to (4) now, it is easier to see how the use of typed feature structures can simplify the specification of transfer rules. Although there are relevant formal differences[15], type definitions and TFS inheritance play a role (abstractly) similar to a macro or template facility. Our correspondence types capture general transfer regularities—direct correspondences and translational divergences alike—and for each such type there can be a potentially large number of specific instances, i.e. actual MTRs with token predicate symbols. Chapter **??** discusses our Norwegian – English transfer grammar in depth, but note that the LOGON project arrived at about 400 abstract correspondence types, with a little over $14,000$ transfer rule instances. Furthermore, the type hierarchy makes it easy to enforce global wellformedness conditions, for example limiting the range of admissible (object-level) variable types (e.g. h, e, and x) and abstractions (i, p, and u, say); enforcing that all relations have a handle (of type h) and predicate (an atomic type, defined in the hierarchy); or that the quantifier-specific

[13] In analogy to a common technique in HPSG, the feature structure formalism would allow the introduction of position-independent references to select elements of the RELS and HCONS lists, say additional features HEAD and MODIFIER for the two externally relevant EPs in our compound example. However, current LOGON transfer grammars have opted for the simpler order coding of such references, so far with no noticeable loss of generality.

[14] The '&' here is the $\mathcal{TDL}$ conjunction operator, and ':=' merely is the syntax for definitions; *seter_mountain+pasture_n* is the identifier (or name) of this rule and is mainly relevant for debugging purposes.

[15] The comparison between TFS systems and the formalism assumed in Lexical Functional Grammar (LFG; see Chapter **??**) is an interesting one, and the HPSG and LFG communities continue to hold mutually incompatible beliefs on the formal differences between the two (somewhat like the communities of C^{++} and Lisp software developers, a cynic might say). LFG assumes untyped structures and instead of TFS inheritance makes heavy use of abbreviatory, parameterized macros. Macros can be recursively defined in terms of other macros, even multiply so, thus achieving effects similar to multiple inheritance.

role labels RSTR and BODY (both with value type *h*) can only occur in conjunction with the inherent argument label ARG0 (typed to *x* in the case of quantifiers).

Summing up our discussion of the TFS to MRS correspondence so far, we employ typed feature structures as *descriptions* of transfer rules, which in turn really are four-tuples of descriptions of MRSs. $\mathcal{TDL}$ is our description language for typed feature structures, and the TFS logic (including its rules of multiple inheritance and TFS wellformedness) implemented in the LKB the primary means of generalization over groups of MTRs. The mapping from a typed feature structure to an MRS is fairly straightforward and discussed in some detail by Copestake et al. (2005) and Chapter **??**. The LOGON transfer formalism extends this correspondence to transfer rules and their added descriptive devices (particularly meta-variables). Assume a complete TFS like the most basic one of Figure 2, i.e. the structure combining information inherited from supertypes with the rule-specific, local specification. The top-level structure has features INPUT, OUTPUT (and optionally CONTEXT and FILTER). Each sub-structure below these features corresponds to one MRS description, with the added constraint that re-entrancies at certain levels are interpreted as transfer meta-variables (just like, ordinarily, TFS re-entrancies are interpreted as logical variables within one MRS). Possible positions for meta-variables correspond to the basic units of MRS contents discussed in Section 3 above: For each EP, these are (i) the handle, (ii) the predicate, and (iii) any of the arguments (values of role labels).[16] Where the TFS contains additional information on these re-entrant nodes, the standard TFS to MRS mapping rules apply. Such information could comprise a specific MRS variable type *e*, say, or constraints on variable properties like $\{\mathsf{TENSE}\ \mathit{past}\}$.[17]

Finally, it is important to observe that the TFS to transfer rule conversion is a *compile-time* operation. Once constructed from their TFS descriptions, MTRs take the form sketched in Section 2 above, viz. as four-tuples of partial MRS descriptions. Accordingly, MTR processing relies centrally on the notions of MRS subsumption and unification defined in Section 3, independent of the TFS type hierarchy.[18]

[16] Although it should in principle, the current implementation does *not* support transfer meta-variables for variable properties, i.e. there is no direct way of creating a *new* object-level variable in the $\mathcal{O}$ component of a transfer rule and let that variable take select variable properties from left-hand side matches of the rule. In practice, we have rarely found such computation necessary, but current LOGON transfer grammars occasionally simulate the intended effect through a combination of transfer-internal EPs and output overwriting; see Section 6 below.

[17] At the implementation level, meta-variables actually use the same internal structure as MRS object-level variables.

[18] The LOGON transfer component to date makes one exception to the separation of the TFS and MRS universes: the hierarchical relations among MRS variable types, values of variable properties, and predicate symbols are represented as part of the standard LKB type hierarchy (note that these types, in all three cases, are atomic, i.e. cannot have internal structure). In principle, however, a good

5 Transfer Ambiguity and Rule Ordering

So far in this chapter we have hardly talked about transfer-level ambiguities or (the closely related topic of) the order of processing in the rewrite system. We defined a transfer grammar as a *sequence* of MTRs, in other words there is a linear ordering to the rules. The order of transfer rules in LOGON is determined simply by the arrangement of MTR descriptions in the source files comprising the transfer grammar. Thus, the transfer grammarian has complete control (and responsibility) over the sequencing of rules; see Section 9 below for alternate approaches to rule ordering. Assume a transfer grammar containing n rules $\langle R_1, \ldots, R_n \rangle$; R_1 will be the first rule to be tried, followed by R_2, and so on. The rewrite process makes a single pass through the MTR sequence—never going back to earlier rules—and terminates when R_n has been processed. Whenever a rule R_i fires, the current MRS M_j is replaced by M_{j+1}, reflecting the effects of invoking R_i; subsequent rule application takes M_{j+1} as its input. Each rule can in principle apply more than once: the original input MRS may contain multiple occurrences of some piece of semantics, for example the relation associated to a preposition that occurred twice in the original source language utterance. Therefore, after R_i has fired, the rewrite process will try R_i again (on M_{j+1} this time), and rewriting will only advance to rule R_{i+1} once the left-hand side condition of R_i (i.e. the totality of its $\mathcal{C}$, $\mathcal{I}$, and $\mathcal{F}$ components) is no longer compatible with the current MRS. This has two logical consequences: (i) rules generating output compatible with their own input requirements can create infinite rewrite cycles; and (ii) for a rule R_i whose input condition occurs multiple times in the current MRS, there can be multiple distinct 'chains' of applications of R_i. For this latter scenario, the rewrite process will explore all such sequences in a pseudo-parallel search; where distinct chains of invoking the same rule multiple times arrive at equivalent results, these would present spurious ambiguities and are thus 'packed' back into a single structure.[19] This property of the rewrite process is discussed further in Section 7 below.

It is of course quite common for an elementary predication or a 'chunk' of connected semantic structure to have more than one possible translational equivalence. Instantiating our basic *noun_mtr* correspondence type once again, examples (11) and (12) show a pair of related transfer rules: both take the nominal relation

part of this specification should be globally hard-wired (by the MRS formalism), and another part be imported from the Semantic Interfaces (SEM-IS; see Chapter **??**) for the source and target language grammars.

[19] For our example of multiple equivalent prepositional relations (call them P_1 and P_2), it would likely not matter in which order the EPs are rewritten, i.e. whether the first application of R_i consumed P_1 or P_2. However, R_i could in principle be sensitive to both P_1 and P_2, for example as part of its $\mathcal{C}$ or $\mathcal{F}$ components; for formal correctness, the rewrite process therefore needs to pursue *all* possible permutations of R_i chains.

associated with Norwegian *hage* as their inputs, where (11) translates it as English 'garden' and (12) as 'orchard'. This pair of rules demonstrates a frequent phenomenon: the target language makes a lexicalized distinction that is not made (in the same way) in the source language.

(11) *noun_mtr* $\wedge \langle \_, \{ \text{\_hage\_n} \}, \{ \} \rangle \stackrel{?}{\rightarrow} \langle \_, \{ \text{\_garden\_n\_1} \}, \{ \} \rangle$

(12) *noun_mtr* $\wedge \langle \_, \{ \text{\_hage\_n} \}, \{ \} \rangle \rightarrow \langle \_, \{ \text{\_orchard\_n\_1} \}, \{ \} \rangle$

One-to-many transfer-level correspondences like this must give rise to transfer ambiguity, i.e. whenever a source language MRS contains the predicate _hage_n, transfer should yield (at least) two output MRSs. To allow such indeterminacy in transfer, we need to augment our formalism. Without extra provision, assuming the rules were ordered like (11) and (12) above, the first rule will fire on all occurrences of _hage_n and always consume its input specification, thus making it impossible for rule (12) to ever be invoked.

The LOGON transfer formalism supports ambiguity of this kind by virtue of *optional* rewrite rules, indicated in (11) through the new '$\stackrel{?}{\rightarrow}$' operator. When rewriting comes to firing an optional rule $R_i^?$ (like (11) above) on the current MRS M_j, the rewrite process forks into two parallel branches: one branch applies $R_i^?$ in the standard way and proceeds to trying $R_i^?$ on M_{j+1}; the other branch skips over $R_i^?$ and advances to trying R_{i+1} on M_j.[20] Each such fork in the rewrite process creates the potential for multiple transfer outputs, where ambiguities that are independent of each other can multiply across branches. Although it is not formally required, it makes sense to mark the last rule in a group of MTRs with identical (or overlapping) input requirements as *obligatory*. This ensures that the source language elements in question are consumed eventually, i.e. it avoids creating transfer branches with 'left-over' SL material.

Groups of related MTRs need not exhibit such perfect parallelism as in examples (11) and (12). The ERG analyzes (the literal sense of) 'go' as intransitive, i.e. _go_v_1 is a one-place relation, while _walk_v_1 constitutes a two-place relation. But the Norwegian *gå* can be used either intransitively or transitively, and transfer needs to make sure to properly match up arguments across translational correspondences. Rules (13) and (14) below differ in their input requirements: (13) uses the special object-level variable type *a* (an 'anti-variable') to prevent unification with a relation that actually has an instantiated second argument. As English 'walk' also has an intransitive use, rule (13) is marked as optional, but there will only be transfer indeterminacy when the input _gå_v occurs as a one-place relation.[21]

[20] In case there were multiple ways of invoking $R_i^?$ on M_j (in the spirit of our earlier example involving multiple occurrences of the same preposition), there will be as many parallel rewrite branches, in addition to the one branch skipping over the rule.

[21] Section 6 below presents another way of achieving the same effect, viz. by changing the con-

(13) $\langle \_, \{ \boxed{h_1}$:_gå_v($\boxed{e_0}$, $\boxed{p_0}$, *a*) $\}, \{\,\} \rangle$
$\overset{?}{\rightarrow} \langle \_, \{ \boxed{h_1}$:_go_v_1($\boxed{e_0}$, $\boxed{p_0}$) $\}, \{\,\} \rangle$

(14) $\langle \_, \{ \boxed{h_1}$:_gå_v($\boxed{e_0}$, $\boxed{p_0}$, $\boxed{x_0}$) $\}, \{\,\} \rangle$
$\rightarrow \langle \_, \{ \boxed{h_1}$:_walk_v_1($\boxed{e_0}$, $\boxed{p_0}$, $\boxed{x_0}$) $\}, \{\,\} \rangle$

In a similar spirit, the pair of rules (15) and (16) have identical $\mathcal{I}$ specifications (though only one of them has a $\mathcal{C}$ component), but they differ substantially in their outputs. The Norwegian *begeistre* conveys variable degrees of effect, which these transfer rules capture as either English 'inspire' or 'fill with enthusiasm'. The latter is analyzed as a two-place relation with an additional predicative set of EPs, parallel to a PP modifier on its second argument.[22]

(15) $\langle \_, \{ \boxed{h_1}$:_begeistre_v($\boxed{e_0}$, $\boxed{p_0}$, $\boxed{x_1}$) $\}, \{\,\} \rangle$
$\rightarrow \langle \_, \{ \boxed{h_1}$:_inspire_v_1($\boxed{e_0}$, $\boxed{p_0}$, $\boxed{x_1}$) $\}, \{\,\} \rangle$

(16) $\langle \_, \{ \boxed{h_0}$:_($\boxed{x_1}$) $\}, \{\,\} \rangle$:
$\langle \_, \{ \boxed{h_1}$:_begeistre_v($\boxed{e_0}$, $\boxed{p_0}$, $\boxed{x_1}$) $\}, \{\,\} \rangle$
$\langle \_,$
$\overset{?}{\rightarrow}$ | $\boxed{h_1}$:_fill_v_cause($\boxed{e_0}$, $\boxed{p_0}$, $\boxed{x_1}$),
$\boxed{h_0}$:_with_p(_, $\boxed{x_1}$, $\boxed{x_2}$), _:udef_q($\boxed{x_2}$, $\boxed{h_2}h$, _),
$\boxed{h_3}$:enthusiasm_n_1($\boxed{x_2}x\{$NUM *sg*$\}$) |
$\{ \boxed{h_2} =_q \boxed{h_3}h \} \rangle$

To summarize our discussion of rule ordering and ambiguity, it should by now be clear that the correct sequencing of MTRs is of central importance. The LOGON transfer formalism shares this property with other resource-sensitive rewrite systems, and our approach is to let the transfer grammar(ian) take complete control over the order of rules. As exemplified by multiple pairs of examples, among groups of MTRs with overlapping input requirements, optional rules must precede non-optional ones. Furthermore, rules with more specific left-hand sides should typically precede more general rules and specifically ones targeting the individual parts of a complex input specification. Going back to our earlier example (3), for

straints on input matching to subsumption (rather than unification) for part of the rule left-hand side.

[22] Chapter **??** discusses the MRS analysis of intersective modification (on the $\boxed{x_1}$ argument in this case, not the event of the relation as a whole): intersective modifiers generally take the index of their modifee as the first argument (ARG1) *and* identify its handle with their own handle (the latter corresponding to logical conjunction in MRS). Rule (16) makes use of its $\mathcal{C}$ component to gain access to the correct handle ($\boxed{h_0}$), i.e. the underspecified context condition is there to 'grab' another variable from the current MRS rather than to constrain applicability of the rule. No matter its predicate symbol, the $\mathcal{C}$ element will need to unify with the relation introducing index $\boxed{x_1}$

the context condition on the cognate object _tur_n to be effective, rule (3) must be ordered prior to any rule that could consume the conditioning predicate _tur_n. LOGON transfer grammars approach the problem of rule sequencing by means of breaking the transfer process into multiple phases, each corresponding to distinct classes of rules. Complex rules like (3), for example, are ordered before purely 'lexical' ones, but after a set of rules that transfer (and to a certain degree harmonize) abstract, structural relations. Note that, except for (3), all examples so far can be viewed as lexical, in that their left-hand sides only make reference to a single specific source language predicate.

6 Fine Points of the Formalism

The preceding sections gave a semi-formal description of the core of the MRS rewrite facilities. The formalism presented this far accurately reflects the original design of the LOGON transfer facilities; however, over the course of about three years—developing transfer grammars of increasing complexity (and other applications of MRS rewriting)—several extensions and one revision were made, most of them to address specialized needs that we did not foresee initially. The following paragraph provide a quick overview of the most important such add-ons.

Skolemization

Subsumption vs. Unification

Bulk Copying

Output Overwriting

Regular Expressions

Variable Property Mapping

7 Implementation Notes — Processing Efficiency

8 Developer Support

9 Related Work

10 Summary — Outlook

References

Carpenter, B. (1992). *The logic of typed feature structures.* Cambridge, UK: Cambridge University Press.

Copestake, A. (2002). *Implementing typed feature structure grammars.* Stanford, CA: CSLI Publications.

Copestake, A., Flickinger, D., Pollard, C., & Sag, I. A. (2005). Minimal Recursion Semantics. An introduction. *Journal of Research on Language and Computation*, *3*(4), 281 – 332.

Fuchss, R., Koller, A., Niehren, J., & Thater, S. (2004). Minimal Recursion Semantics as dominance constraints. Translation, evaluation, and analysis. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics.* Barcelona, Spain.

Krieger, H.-U., & Schäfer, U. (1994). *TDL* — A type description language for constraint-based grammars. In *Proceedings of the 15th International Conference on Computational Linguistics* (pp. 893 – 899). Kyoto, Japan.

Rounds, W. C., & Kasper, R. T. (1986). A complete logical calculus for record structures representing linguistic information. In *Proceedings of the 15th Annual IEEE Symposium on Logic in Computer Science.* Cambridge, MA.