

Towards a General Framework for Underspecification in NLG and an Underspecification Language for MRS

COGENT Project Report
Proposal for Work Package 2: “Design of Abstract MRS”

Anja Belz

16 April 2004

Contents

1	About this document	3
2	WP2 in the context of COGENT	3
2.1	COGENT	3
2.2	A new generation environment as research platform for COGENT	3
2.3	Requirements for AMRS	4
3	Summary of MRS	5
3.1	Elementary predications	5
3.2	MRS	6
3.3	MRS with QEQ-constraints	6
3.4	MRS with QEQ-constraints and the Lingo/ERG composition algorithm	7
3.5	Summary and comments	7
4	Nondeterminism in the Lingo/ERG Generator	7
4.1	Word order	8
4.2	Other syntactic nondeterminism	8
4.3	Lexical nondeterminism	8
4.4	Underspecification of variable features	9
4.5	Underspecification of relations	10
4.6	Unclassifiable forms of nondeterminism	10
4.7	Summary	10
5	Underspecifiable representation in NLG: Towards a formalism	11
5.1	Some basic choices	11
5.1.1	MRS and Lingo/ERG as it is?	11
5.1.2	Use an underspecification formalism unrelated to MRS?	11
5.1.3	Half-way house?	12
5.1.4	Do we need an explicit more abstract level of representation?	12
5.2	Generation space and intermediate representations	13
5.3	Underspecifiable representation spaces	14
5.4	AMRS as modelling a generation space decision tree	15
6	Next steps: Implementing the underspecification framework and the AMRS language	16
6.1	Framework for representational underspecification	16
6.2	AMRS Language	17

1 About this document

This document is a COGENT project report and presents a proposal for COGENT work package WP2 (“Design of Abstract MRS”). It is organised as follows. The first section below outlines the role WP2 plays in COGENT. This is followed in Section 3 by a summary of the Minimal Recursion Semantics (MRS) formalism [Copestake et al., 1999, Copestake et al., 2003]. Section 4 describes how the Lingo/ERG generator works, and discusses the extent to which nondeterminism is already present in it. Section 5 discusses some basic choices for WP2 and outlines one possible view of the generation process and the possible role of AMRS in it. The final section outlines the next steps for research in COGENT, towards the full formal specification of a generalised representational underspecification framework for NLG, and a specific underspecification language for MRS implemented in the framework.

The main part of this report was written at the end of 2003 and the beginning of 2004, with minor changes made at the end of 2005.

2 WP2 in the context of COGENT

2.1 COGENT

The purpose of COGENT is to look at issues in wide-coverage generation, in particular the issue of nondeterminism. Nondeterminism arises from two main sources in NL generation: (i) **wide coverage**: the wider the coverage of grammar and lexicon, the more word strings can be generated from the same semantic representation, for any given semantic representation formalism¹; and (ii) **underdetermined semantics**: the less specific the semantic representation, the more word strings correspond to it.

Having wide coverage and permitting (even extensively) underdetermined semantics are both desirable in certain circumstances: both are important issues in generator portability and reusability. However, at present there is no comprehensive methodology for **controlling the resulting nondeterminism**², for choosing, in other words, among several different possible realisations.

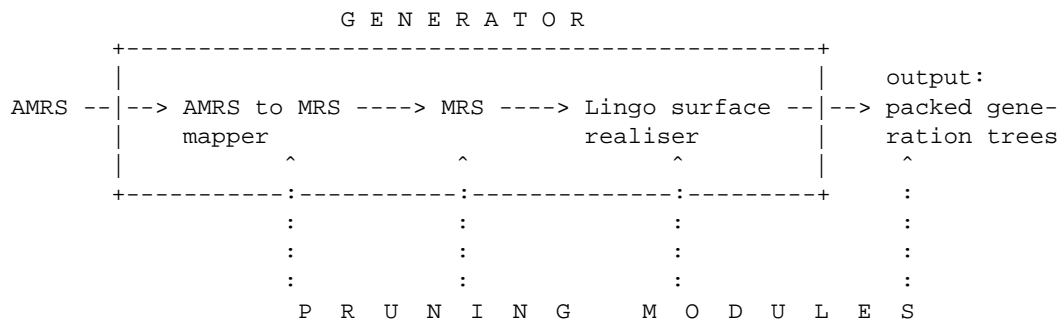
The core aim for COGENT is to develop methods for controlling nondeterminism in the mapping from semantic representation to surface realisation. Work towards this aim breaks down into two separate steps: (i) creating a nondeterministic generation environment, and (ii) developing methodologies for pruning the generation space.

2.2 A new generation environment as research platform for COGENT

In the first project phase we aim to create a generation environment as a research platform for COGENT researchers. This involves (i) **designing a semantic representation formalism**, and (ii) **creating a wide-coverage generator** capable of generating from the inputs represented in the formalism. We will also (iii) **create corpora of semantic representations** to use as inputs to the generator (also to be made available to other researchers).

As part of the generator we plan to use an existing (surface) realiser (Lingo) which requires highly specified, lexicalised input, and has very little nondeterminism. So the plan for WP2 was to design a semantic representation formalism (working title: AMRS, as in *Abstract MRS*) that allows substantially more semantic underdetermination than MRS. At the least such a formalism needs to define a mapping from more abstract semantic relations to the strongly lexicalised semantic relations of Lingo/ERG MRS.

The generation environment will have the following general modular architecture, where the modules for pruning the generation space (“pruning modules”) can potentially be applied in four different places — either operating on output data structures (MRS, packed generation trees), or inside procedures (AMRS/MRS mapping, chart generation):



3 Summary of MRS

This section is a summary of the main features of the MRS formalism as described in the 2003 version of the introductory paper on MRS that's been available as a manuscript for some years [Copestake et al., 2003] The summary is somewhat biased in the direction of relevance to the COGENT project.

For the sake of what I hope is increased clarity, I put things differently (from the Introduction paper) in some places in this summary, in particular:

- the paper is a bit confusing about handle terminology, not distinguishing handles that label an EP from handles that are arguments in scopal relations. I use the word **slot** for what's mostly called a handle argument in the paper, and the word **label** for what's variously called a handle or a label (among other things) in the paper
- the incremental presentation in the introductory first half of the paper is misleading in some places (they say things that are correct for the intermediate stage at which they are being said, but not for MRS in its final, complete form). The summary below is non-incremental.
- I say that an EP is a relation and a label, and a relation has a name and arguments, rather than saying that an EP has the four components handle, relation, list of variable arguments, list of handle arguments (p. 12)
-

MRS is a representational framework for defining a metalanguage for representing and (structurally) underspecifying trees in some given object language (OL) of trees.

MRS is not a completely general tree underspecification framework: it is not the case that there is an underspecified MRS expression for every subset of trees in the object language.

Under the **general formulation**, there is an MRS for every subset of trees that have the same set of nodes. Under **MRS with QEQ-constraints** there is an MRS for every subset of trees that have the same set of nodes *and* are structurally identical except for quantifier attachment.

Under **MRS with QEQ-constraints and the Lingo/ERG composition algorithm**, there is an MRS for every set of *all and only* those trees in the object language that have the same set of nodes and are structurally identical except for quantifier attachment. (I.e. there is no MRS for any strict subset of such a set.)

Each of these three, increasingly restrictive formulations will be described in slightly more detail in the remainder of this section.

3.1 Elementary predications

“The underlying assumption behind MRS is that the primary units of interest for computational semantics are *elementary predications* or EPs, where by EP we mean a single relation with its associated arguments (for instance, *beyond(x, y)*). In general, an EP will correspond to a single lexeme.” (p. 3)

An EP consists of a relation together with a unique label. There are three basic types of relation:

1. *non-scopal relation*: arguments are variables only, e.g.: *cat(x)*, *see(x, y)*.
2. *fixed scopal relation*: arguments may be variables and/or slots, e.g. *probably(s)*, *or(s1, s2)*.
3. *generalised quantifier relation* (aka *floating scopal relation*, or simply *floating relation*): first argument is a variable, second and third arguments are slots; *relation(variable, restriction, body)*, e.g. *some(x, s1, s2)*.

There are at least two special cases of fixed scopal relations:

1. p. 31 *message relations*: $l : m_rel(s)$; can't do $srel(...s_a...)$, $s_a =_q l$, but must do $srel(...l...)$ directly; so that a message relation can't itself introduce new scope possibilities; e.g. *prpstn_m_rel* for propositions, *int_m_rel* for questions. All complete underspecified MRSs have an outermost *m_rel* EP.
2. p. 40-41 *coordination relations*: $l : c_rel(ls, rs)$; *ls, rs* cannot form lhs in a $=_q$ relation, but are simply equal to the labels of the coordinated constituents; effect: no (additional) quantifiers can come between; every list and other sequence of coordinate constituents is done as a binary tree of *c_rels*

variables are the same as in ordinary logic representation, e.g. x is a variable in $\forall x : bird(x) \rightarrow has_wings(x)$.

slots are arguments in scopal relations and can be in qeq relations (in underspecified MRSs) and equality relations (in fully specified MRSs) with labels

labels are components of EPs, each EP has exactly one label, within an MRS, all EPs with distinct labels must attach to different slots, while all EPs with identical labels must attach to the same slot (where to be attached means to be in either a qeq or an equality relation).

3.2 MRS

Given some object language (OL) of trees T , the MRS metalanguage for T , denote it M_T , consists of all and only those expressions that can be derived from an expression of T in the following way (I'm assuming that nodes representing logical conjunction, 'conjunction nodes', if any, are identifiable as such)³.

1. except for child nodes of conjunction nodes, give all nodes in the tree a unique label
2. starting at the tree root, label all child nodes of conjunction nodes with the label of the parent node; then remove the parent node and replace it with the bag of all labelled child nodes $\{|l : C_1, l : C_2, \dots, l : C_n|\}$
3. starting at the tree root, for each labelled node $l:N(x_1, x_2, \dots, x_n)$ where the x_i are any arguments the node may already have create an elementary predication (EP) $l:N(x_1, x_2, \dots, x_n, s_1, s_2, \dots, s_m)$, where N has an additional argument s_i for every unique label s_i among the child nodes of N ; l is the **label** of the EP, the x_i are the variable arguments, and the s_j are slot arguments
4. create the bag E of all EPs created in last step; let GT be equal to the label of the root node; then $\langle GT, E, \{\} \rangle$ is a fully specified MRS expression
5. replace any number of slot arguments in E with unique new ones (this has the effect of cutting — underspecifying — a parent node / child node connection)
6. let C be some language of tree constraints and let $D \subset T$ contain all and only those elements of T from which E can be derived by following steps 1–5 above; let $c \in C$ be some representation of tree constraints that, in conjunction with GT and E can be used to derive one of the subsets of D ; then, $\langle GT, E, c \rangle$ is a well-formed underspecified MRS expression

An MRS in which no parent/child node connections are underspecified is called a scope-resolved MRS (here: **fully specified MRS**). All other MRSs are called **underspecified MRSs**.

Representational power: There exists a fully specified MRS for every expression in T . But it is not the case that there is an underspecified MRS for every subset of T . Let T_{N_1, N_2, \dots, N_n} denote a subset of T containing all and only trees with the set of nodes N_1, N_2, \dots, N_n . Then for every $D \subseteq T_{N_1, N_2, \dots, N_n}$ there exists an MRS M such that M can be derived (by following the above steps) from all and only the elements of D . In other words, there is an underspecified MRS for any set of trees in T that have the same node set. There are no other MRSs.

3.3 MRS with QEQ-constraints

Let T again be the OL, M_T the MRS metalanguage for T , $M = \langle GT, E, C \rangle$ an expression in M_T , and D the set of expressions in T from which M can be derived by the steps above. QEQ-constraints provide one specific way of identifying a subset of D .

Under MRS with QEQ-constraints, C is a set of pairs $l =_q s$ where l is a label on one of the EPs in E , and s is a slot in one of the EPs in E . The meaning of the constraint is that $l =_q s$ (read: l is q e q to s) must be true for l and s in E . $l =_q s$ is true either if $l = s$, or if l is the label of an EP that dominates the EP that contains s , and at the same time all other EPs that are dominated by l and that dominate s are quantifier EPs. In other words, if anything comes in between l and s in the tree structure then it must be a quantifier EP.

QEQ-constraints give a subset of EPs — the quantifier EPs — a special status. The idea is that quantifier EPs correspond to quantifier relations in OLs that are generalised quantifier logics. However, the only formal requirement is that the set of quantifier EPs in M_T is identifiable somehow, so that $l =_q s$ can be evaluated.

The effect of using QEQ-constraints is that there is no longer an MRS for every subset of every $T_{N_1, N_2, \dots, N_n} \subseteq T$ (using the same notation as in the previous section). The underspecified MRS for some given T_{N_1, N_2, \dots, N_n} itself has an empty set of QEQ-constraints. Every underspecified MRS with a non-empty set of QEQ-constraints picks out one of those subsets of T_{N_1, N_2, \dots, N_n} in which all trees have the same QEQ-relations. In other words, there is an MRS only for those subsets of T_{N_1, N_2, \dots, N_n} whose elements share the same QEQ-relations (possibly none), which means having the same nodes **and** the same structure except for quantifier attachment.

³Because the OL is a tree language, I'm further assuming that a few things can be done that can usually be done with trees, such as that the set of nodes and parent/child relationships can be determined and that the tree can be traversed top down.

3.4 MRS with QEQ-constraints and the Lingo/ERG composition algorithm

Intuitively, the effect of the Lingo/ERG composition algorithm is to construct two types of trees: for each MRS, one so-called **backbone tree** and several quantifier trees. The algorithm fixes the tree structure for all non-quantifier EPs in accordance with the QEQ-constraints in the MRS. The resulting tree is called the backbone tree. For every quantifier EP, the algorithm attaches EPs to the slot that corresponds to the restriction of the quantifier, and leaves all other connections (including the body of the quantifier) unrestricted.

That's the only kind of underspecified MRS that can be constructed in Lingo/ERG. The fully specified forms are then derived by inserting the quantifiers in all possible ways into the backbone tree, where a quantifier can be inserted *between any parent and child node in the backbone tree*⁴.

From the perspective of the OL, this means that under MRS with QEQ-constraints and the Lingo/ERG composition algorithm there is an MRS only for those subsets of T_{N_1, N_2, \dots, N_n} (using the same notation as in the previous sections) whose members are structurally identical except for the location of the quantifier nodes.

3.5 Summary and comments

The three MRS formulations outlined in this section differ only in terms of the set of underspecified MRSs they allow. MRS with QEQ-constraints and the Lingo/ERG composition algorithm is very restrictive. Even though general MRS allows all kinds of underspecification, with Lingo/ERG composition, only quantifier scope can be underspecified, moreover, it can only be underspecified in one very specific way: everything except quantifier attachment must be specified, while quantifier attachment and quantifier bodies must not be restricted in any way⁵.

This has the significant result that even if the syntactic position of the quantifier lexeme (partially) disambiguates quantifier scope, the Lingo/ERG parser has no way of taking this into account — it must pretend that the position of the quantifier lexeme has no disambiguating effect.

Example: to both sentence (1) and sentence (2) below, Lingo assigns both readings (3.1) and (3.2)⁶:

1. *some dogs only bark*
2. *only some dogs bark*
- 3.1 `only(some(dogs,bark))` as in: only some dogs bark (but not some cats bite).
- 3.2 `some(dogs,only(bark))` as in: some dogs only bark (but don't also bite).

(3.2) is a valid reading for (1) but it isn't for (2).

There's another problem. Lingo doesn't get a third possible reading at all:

- 3.3 `some(only(dogs),bark)` as in: only some dogs bark (but not wolves).

The reason for this is equally fundamental: scopal relations that are not quantifiers can never end up taking scope over any part of the restriction of a quantifier (unless they are in the body of another quantifier nested inside the restriction). This is because underspecified MRSs consist of a set of quantifiers and a tree of the remaining EPs that has marked slots where the quantifiers can be inserted. A quantifier can be inserted either above a non-quantifier scopal relation (in which case the latter ends up in the body of the quantifier), or below it in which case the entire quantifier ends up in the scope of the non-quantifier scopal relation.

4 Nondeterminism in the Lingo/ERG Generator

The Lingo/ERG generator is not the exact reverse of the parser. It takes an MRS structure as its input, but because it's a (lexicalist) chart generator, it ignores parts of the input MRS, most importantly it doesn't look at the labels and slots. Instead it generates all possible ways in which the EPs can be put together, and then — during a postprocess — it checks the label and slot instantiations in the outputs against the QEQ-constraints and against those in the input MRS. So it's at least in part a generate-and-test situation.

Individual generator steps:

1. lexical lookup: map relations to inflected lexical items
2. chart generation: find all ways according to grammar in which lexical items can be put together to form a sentence; for each sentence, the syntactic structure and the complete MRS are constructed in the same process

⁴Only two types of EPs are mentioned in the Introduction to MRS paper as exceptions.

⁵I think this treatment of quantifiers is in fact equivalent to nested Cooper-storage, but I haven't read enough about Cooper-storage to say for certain at this point.

⁶For a different syntactic analysis, Lingo also assigns the reading `only(some)(dogs,bark)` as in: only some dogs bark, not all.

3. equivalence check: for each sentence generated, check its MRS structure against the original input MRS for equivalence; currently, this appears to be using a fairly loose definition of equivalence

The remainder of this section looks at the different kinds of nondeterminism in the Lingo ERG generator.

4.1 Word order

Since generation is from *bags* of EPs, in principle lexemes can be realised in any order allowed by the grammar. For example generating from the MRS for *the bad sad dog barked* will produce both of the following:

```
the bad sad dog barked
the sad bad dog barked
```

However, this doesn't work when the adjectives are explicitly coordinated. E.g. the MRS for *the bad and sad dog barked* only generates:

```
the bad & sad dog barked
the bad and sad dog barked
the bad and then sad dog barked
```

Furthermore, this doesn't work for noun modifiers, lists of nouns, lists of verbs, and works unpredictably for prenominal modifiers that aren't all adjectives. In fact, the only examples I could find for which it works are lists of adjectives or adverbs (as long as there is no coordination).

4.2 Other syntactic nondeterminism

There isn't a lot of room for alternative syntactic constructions, since generally every word that appears in the realisation must have its corresponding semantic relation present in the input MRS, except for a few well-defined exceptions.

Even the example from the COGENT proposal *wipe the dropper's end/wipe the end of the dropper* can't be generated from the same MRS (at least not with the current Lingo/ERG). For *of* to appear in the realisation, `_of_rel` has to be in the input MRS.

Syntactic variation seems mostly to be limited to word order, as described in the previous section. Other examples I found are:

- verb forms, e.g. the MRS for *help him read* generates:

```
be helped him to read
help him read
help him to read
help him read
help him reading
```

- order of phrases: e.g. *it's good to talk* generates both *to talk it's good* and *it's good to talk* (as well as many other verb form variants), but not e.g. *talking is good*.
- position of adverbs seems to be pretty free, e.g. from the MRS for *carefully wipe it* you get both possible orderings:

```
carefully wipe it
wipe it carefully
```

Additional syntactic nondeterminism can in principle be achieved by underspecifying certain relations (see Section 4.5 below).

4.3 Lexical nondeterminism

There are instances of multiple entries in the lexicon that have the same semantic relation, although these are hard to find and tend to concern spelling variants:

```

am_temp := xp_am_pm_le &
  [ STEM < "a.m." >,
    SYNSEM.LOCAL.KEYS.KEY _am_rel ].
am_temp_2 := xp_am_pm_le &
  [ STEM < "am" >,
    SYNSEM.LOCAL.KEYS.KEY _am_rel ].

```

```

a-det := det_sg_nomod_le &
  [ STEM < "a" >,
    SYNSEM.LOCAL.KEYS.KEY _a_quant_rel ].
an := det_sg_nomod_le &
  [ STEM < "an" >,
    SYNSEM.LOCAL.KEYS.KEY _a_quant_rel ].

```

The treatment of synonymy in the Lingo/ERG lexicon seems to be very conservative, generally there is one relation for one lexical item (where a lexical item may be more than one word). Other examples I found are:

- the pairs somewhere/someplace and anywhere/anyplace which are analysed as `place_rel`, `_some_rel` and `place_rel`, `_any_rel` respectively
- to/in order to:

```

in_order_to := p_subconj_inf_le &
  [ STEM < "in", "order", "to" >,
    SYNSEM.LOCAL.KEYS.KEY _in_order_to_rel ].

```

```

to_subord := p_subconj_inf_le &
  [ STEM < "to" >,
    SYNSEM.LOCAL.KEYS.KEY _in_order_to_rel ].

```

- `numbered_hour(x15, 12, v20, v19, v21)` will generate *noon*, *midday*, *noontime*, *twelve*.

Additional lexical nondeterminism can in principle be achieved by underspecifying certain relations (see Section 4.5 below).

4.4 Underspecification of variable features

Each variable argument (e.g. event variables, object variables) in an MRS representation comes with a set of feature/value pairs. A hierarchy is defined for the feature values. In principle, any variable feature can be underspecified by selecting a value that isn't right at the bottom of the hierarchy.

E.g. event variables have a feature for tense. Possible (fully-specified) values are FUTURE, PRESENT and PAST. If a value higher up in the hierarchy is chosen instead, a larger set of sentences can be generated:

```

e2 [ EVENT
    DIVISIBLE:  BOOL
    E.MOOD:     INDICATIVE*
    E.TENSE:    TENSE
    E.ASPECT:   PROGR* ]

(("he" "is" "sleeping") ("he" "s" "sleeping") ("he" "'s" "sleeping"))
("he" "was" "sleeping") ("he" "will" "be" "sleeping")
("he" "ll" "be" "sleeping") ("he" "'ll" "be" "sleeping")
("he" "shall" "be" "sleeping") ("he" "is" "gonna" "be" "sleeping")
("he" "s" "gonna" "be" "sleeping") ("he" "'s" "gonna" "be" "sleeping")
("he" "was" "gonna" "be" "sleeping")

```

But this interacts with a filter for lexical items that don't have corresponding semantic relations. The filter consists of rules such as 'introduce auxiliary 'had' if the tense in the input is past perfective'. However, this means that an underspecified tense can never get you the past perfective (because the verb will only be realised in the past perfective if the semantics *specifies* the tense as past perfective)⁷.

In the current version, furthermore, either the filter doesn't have enough rules in it or something else goes wrong when features are taken into consideration during generation. E.g.: `BOOL:MODAL_SUBJ*:FUTURE*:PROGR+PERF` is the verb analysis for *he would have been sleeping*, but if you use it to generate, you get *he would have slept* (and variants), but not *he would have been sleeping* (or any variants).

⁷I tried this out with different underspecified values for the tense feature, and the results included a range of different tenses, but not past perfective.

4.5 Underspecification of relations

Relations too are part of an underlying type hierarchy in Lingo/ERG. In principle, relations can be underspecified by using any supertype from the hierarchy. The example discussed in [Carroll et al., 1999, p. 88] is prepositions with temporal location senses, where one might want to use the supertype ‘temporal location’ in order to leave the choice between prepositions to the grammar. E.g. *on Tuesday morning* vs. *in the morning on Tuesday*. Lingo/ERG has the following:

```
_at_temp_rel :< temp_loc_rel
_in_temp_rel :< temp_loc_rel
_on_temp_rel :< temp_loc_rel
```

Where the temporal location senses of *at*, *in*, *on* are subtypes of the same general type `temp_loc_rel`. So in principle it is possible to use `temp_loc_rel` in the input MRS and to let the grammar choose which of `_at_temp_rel`, `_in_temp_rel`, `_on_temp_rel` to use in the realisation.

However, this possibility in principle interacts in practice with the way lexical items are selected for chart population. Before the Lingo/ERG generator can be run, the **indexing procedure** has to be run. Lexemes are pre-indexed and then retrieved at generation time based on the indexed type. The lexemes are supposed to be indexed by supertype as well as their own type, so that if you put in an MRS containing a supertype, a set of lexemes will be retrieved. But there is a limit on the indexing — the code doesn’t support generation from completely underspecified MRSs because of efficiency issues.

Looking at the hash table created by the indexing procedure, I haven’t been able to find a single example of a lexeme indexed both by its own lexical type as well as a supertype. Either the relevant code isn’t working⁸ or the limitation on supertype indexing is very severe.

4.6 Unclassifiable forms of nondeterminism

The (unambiguous) MRS for *dogs do not bark* generates all of the following:

```
dogs do n’t bark
dogs do not bark
dogs do not bark
dogs do not really bark
dogs do not really bark
dogs do still not bark
dogs do still not bark
dogs don t bark
```

Apparently any coordinated list of adverbs or adjectives permits the introduction of ‘then’ after the conjunction. E.g. the MRS for *the mad, sad and bad dog barks* generates (after about 10mins):

```
the mad, sad & bad dog barks
the mad, sad and bad dog barks
the mad, sad and then bad dog barks
```

4.7 Summary

The Lingo/ERG generator is a lexicalist chart generator. It generates from bags of lexical items and must use a postprocess to ensure that handle constraints in the input MRS are observed.

Both the grammar and the lexicon are nondeterministic to some extent. The examples above show that syntactic nondeterminism is somewhat unpredictable. Furthermore, there appear to be mechanisms that aren’t part of the grammar but restrict syntactic nondeterminism.

In addition, the generator has an underspecification mechanism for variable features which can be underspecified by using feature values higher up in the feature value hierarchy. However this isn’t working properly at the moment (see example above).

Since there is also an underlying hierarchy of types of semantic relations, it’s in principle possible to underspecify relations by using relations higher up in the relation type hierarchy. However, this isn’t currently possible and it looks as though substantial work would be needed to make it work.

⁸Ann says it’s quite possible this part of the code isn’t doing what it’s supposed to with the current grammar.

5 Underspecifiable representation in NLG: Towards a formalism

The purpose of this section is to outline a basis for AMRS. It starts by discussing some preliminary alternatives (Section 5.1). Next it takes a high-level look at the entire generation process and in particular the role of semantic representations (of which MRS is a highly specified example), to get an idea of the possible as well as the desirable roles played by AMRS and MRS within this larger context (Section 5.2).

The next section (5.3) compares the representational spaces covered by conventional and underspecifiable semantic representation languages. The underspecifiable representation space is described in terms of a tree where the leaf nodes are the fully specified representations and the other nodes are nonspecific between the leaf nodes they dominate. It is argued that such a representational space can be used to explicitly model (part of) the decision tree underlying the generation process as a whole.

Section 5.4 argues that this is precisely the way AMRS should be viewed: as an underspecifiable representation formalism that models the lower part (or an arbitrarily low part) of the decision tree underlying the generation space as a whole. The proposal is therefore to implement AMRS as a generic framework with a nongeneric component that defines the ways in which representations can be underspecified, such that generic and nongeneric components have much the same roles as parser and grammar in NLU.

5.1 Some basic choices

5.1.1 MRS and Lingo/ERG as it is?

One possibility is to use and/or extend the capabilities for nondeterminism already present in Lingo/ERG, without creating external mechanisms for increasing nondeterminism. If this meant less work and would let us look at all the things we want to look at, it would be tempting.

There are a number of reasons why it isn't feasible though:

- the semantic relation type hierarchies in Lingo weren't created with underspecification in mind and many types are not meaningful in terms of semantic representation; there is no simple way of excluding such types from the part of the hierarchy that is used for underspecification
- the Lingo/ERG type hierarchies are unlikely to cover everything COGENT wants to look at (for example there is no common supertype for active/passive constructions, either in feature values or semantic relations), so we'd have to find a way of extending the hierarchies for underspecification, while at the same time preventing other parts of the Lingo parser/generator from using the dedicated underspecification types
- other Lingo mechanisms interact with the built-in possibilities for underspecification and mean it doesn't quite work the way you'd expect; repairing this would be a lot of work
- Lingo/ERG is work in progress — we want to benefit from further development, which will not automatically be possible if we change it (at the same time, further development may result in features we don't want to use)
- Lingo/ERG isn't ours to change arbitrarily
- a representation language that isn't identical to Lingo/ERG MRS comes with the possibility in principle to link up with other surface realisers on the one hand, and a range of different reasoners on the other

5.1.2 Use an underspecification formalism unrelated to MRS?

Given the various current shortcomings of MRS (unpredictable underspecification mechanisms, no formal underpinnings, no semantics, unpublished, representational limitations), we need to consider opting for a separate semantic representation formalism (SRF), not necessarily related to MRS, that has sounder foundations and is more representationally adequate for our purposes. But:

- the mapping from SRF to MRS would become more difficult to define
- would need to find a suitable existing SRF — but what would be our criteria for choosing one, apart from ease of mapping to Lingo/ERG MRS?
- it might not be possible to incorporate future extensions to MRS and Lingo/ERG into the mapping in a straightforward way
- future grammar/lexicon development in ERG may also make nontrivial changes to the mapping necessary.

5.1.3 Half-way house?

There are several good reasons why we *do* want to use Lingo/ERG:

- it's one of the few freely available wide-coverage realisers
- ERG will be continuously updated and improved
- MRS taps into a fairly large existing research community
- saves us work
- it's reversible (does both parsing and generating with the same grammar)

The best strategy for AMRS lies somewhere in between the two considered above: a formalism closely enough related to MRS to make the mapping easy, yet separate enough from Lingo/ERG to make it possible to work around its various drawbacks.

It's probably a good idea — in view of the drawbacks and the fact that it's work in progress — to encapsulate Lingo/ERG as far as possible, and not to rely on anything except the fact that it can take fully specified MRS representations as input. This would also keep open the option of switching to a different MRS-based grammar/generator.

One part of ensuring encapsulation is to ignore the (unpredictable and fragile) mechanisms that Lingo/ERG has for underspecifying features and relations and pass only fully specified representations to it. It will probably also be necessary to change the equivalence check that Lingo/ERG performs at the end of generation, to incorporate a stricter notion of equivalence. E.g. to avoid insertion of 'then', 'really' and similar (see Section 4.6 above).

The one part of Lingo/ERG that can't be ignored is its set of lexical semantic relations and variable features, as fully specified Lingo input has to be composed of these. The interface between AMRS and MRS needs to be defined clearly and cleanly in terms of this set of lexical semantic relations and the ways in which they can combine.

5.1.4 Do we need an explicit more abstract level of representation?

So far, this report has considered increasing nondeterminism in the generation system in terms of creating a new level of representation that is in some sense more abstract than Lingo/ERG MRS and therefore is in a 1-to-many relation with it. Under this view, the inputs to the COGENT generation system would be encoded in the more abstract representation language.

A conceivable alternative is *not* to have a new explicit level of representation at all, but simply to map directly from one MRS representation to a set of MRS representations, by means of a set of 'transformation rules'. One advantage would be that we wouldn't need to worry about how to create our corpora, as these would simply be files of MRSs.

The situation would be something like this. We would need to define sets of equivalent MRS representations $M = M_1, M_2, \dots, M_n$. There would be a set of transformation rules to derive for each M_i all other elements of M .

But:

1. this would mean having to define a normal form (NF) representation as well as a set of (ordered) transformation rules - which is not that different with regard to development effort and possibilities in terms of representational power from defining an explicit new representation formalism that is then mapped to MRS
however, it comes with the significant disadvantage that a NF representation would be a fully specified MRS too, and would contain no information about the ways in which it is 'underspecified'; every explicit underspecification formalism does contain this information
2. there would be no way of distinguishing fully specified representations from (partially) underspecified representations
3. there could only be two levels of representation: on the one hand, the set of equivalent representations (or, equivalently, the NF representation), and on the other hand, the individual representations
a new representation formalism can be defined in such a way that different levels of specificity can be encoded (picking out different-sized sets of MRSs), and a specificity hierarchy defined over individual representations
4. when interfacing with other reasoners (which we don't want to rule out completely) what would the inputs be? do we want to require them to provide fully specified MRS input? requiring highly specified inputs is one drawback of available wide-coverage realisers we've been criticising
5. whether generating from our corpora or taking input from an external reasoner, there would be a peculiar mismatch between the input and the outputs, e.g. an input specifies passive and the set of outputs includes both actives and passives

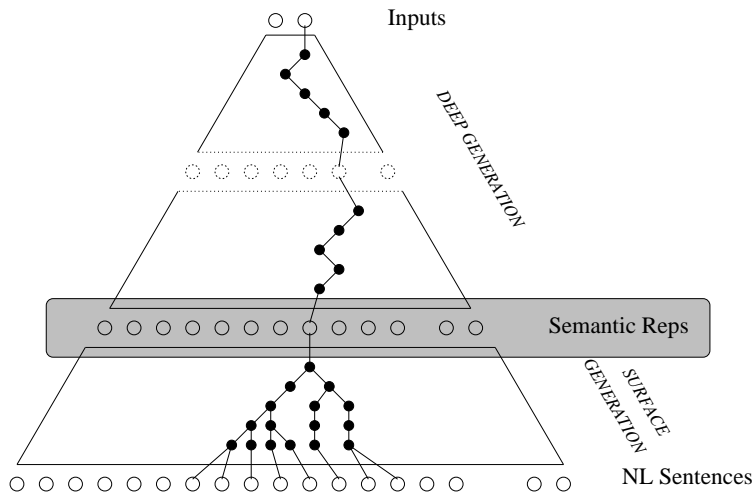


Figure 1: Generation space as decision tree.

5.2 Generation space and intermediate representations

While there is a lot of disagreement in the NLG community about appropriate architectures, interfaces and the representations that are constructed, there is broad agreement that the NLG process passes through three phases, roughly corresponding to three levels of representation: (1) conceptual, (2) semantic, and (3) syntactic and lexical (even [Reiter, 1994] and the RAGS project [Cahill et al., 2001] agree in these general terms). This is not to say that all three phases are completely separate — in the sense that there is a well-defined interface with an associated representation language — in all NLG systems, but most have at least a distinct level of representation that is intermediate between conceptual representation and NL strings (i.e. a semantic level, in the broad sense).

Figure 1 is an illustration of the generic three-phase generation space. The unfilled circles are data structures representing intermediate representations. From one level to the next, representations become more specific in terms of determining the NL string. The filled circles in the figure represent decision nodes, and the idea is that any generation space can be equivalently represented as a decision tree⁹ (in this figure, strictly speaking, a set of decision trees). From the input to the NL output a chain of decisions are made that determine the paths that are taken through the generation space, and ultimately the output set of NL strings. Decisions can be implicit or explicit, based on properties of intermediate data structures or on generation context. The generation system may have partial or additional intermediate representations corresponding to some of the decision nodes, but more often it doesn't. There may be many different ways of representing a given generation space as a decision tree, the idea is not that there must be an identifiable data structure corresponding to each node. For example, in a system capable only of generating active constructions, the decision between active and passive is implicit, and an equivalent decision tree space may or may not contain a decision node representing the 'decision' in favour of active.

As an example of an intermediate, conceptual-level representation, consider the following representation from [Reiter and Dale, 1997, p. +9]:

```

+-
| message-id:msg02
| relation: DEPARTURE
|
|           +-
|           | departing-entity: CALEDONIAN-EXPRESS
| arguments: | departure-location: ABERDEEN
|           | departure-time: 1000
|           +-
+-
+-
+-

```

At some point between this representation and the set of NL strings that can be generated from it in a given system (e.g. *The Caledonian Express departs from Aberdeen at 10am*, *The train departs from Aberdeen at 10am*, *It leaves Aberdeen at 10am*, *It leaves here in 5 minutes* etc.) a large number of decisions have to be made. Some of these are conventionally dealt with by a referring expressions module, some by a process of lexicalisation, some by the grammar. An example of an individual, atomic decision (that would be a single decision node in the graph in Figure 1) is that between the verbs 'depart from' and 'leave'. Depending on the system, the decision may be implicit in the

⁹There will tend to be several ways of getting to the same (intermediate) representation, so the data structures aren't meant to be unique. But because I'm considering the generation space strictly from the generation point of view here, it's more convenient to represent it as a tree.

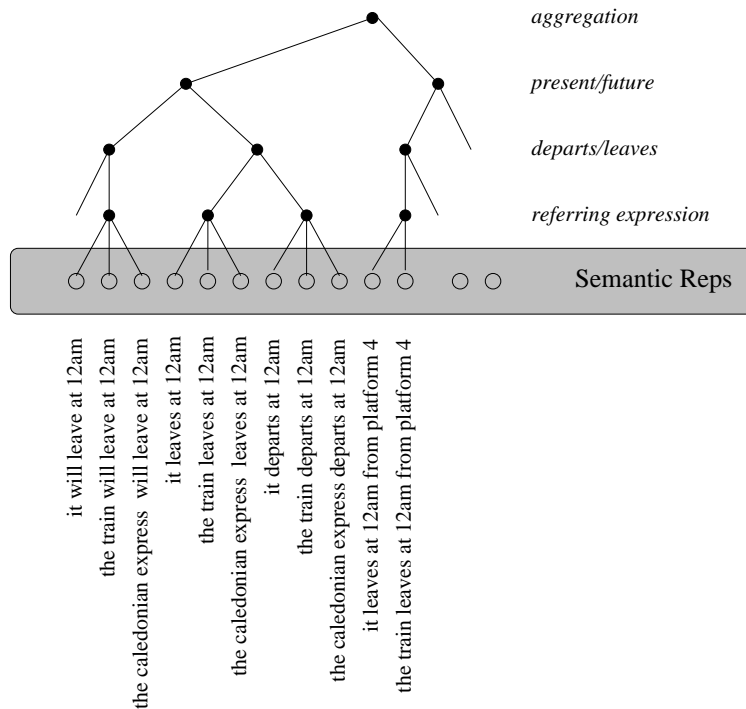


Figure 2: Underspecifiable representation space.

conceptual-level representation, so that only word strings containing, say, ‘leaves’ are accessible from it, or there may be a point in the generation process where the two alternatives are explicitly considered and a decision is made based on some information external to the representation. A third option is that both paths are pursued and the decision is postponed (so that if and when made it will cut off all but one of the paths). Multiple paths being pursued means nondeterminism in the system.

The connections between decision nodes in Figure 1 show a typical scenario in a generation system: in the conceptual and semantic space, decisions tend to (explicitly or implicitly) lead down single paths, but the grammar and/or lexicon has some degree of nondeterminism. Often, all possible realisations are generated and a single one is selected by a postprocess (which often simply selects the first string generated).

Seen as a whole, the space in Figure 1 represents all possible individual generation processes in a system, i.e. all the different ways you can get from an input to an output. An individual generation process is a path or multiple paths through this space.

NLG systems tend to have two distinct formalisms for intermediate representations roughly corresponding to a conceptual and a semantic level (as indicated in the figure). These tend to correspond to interfaces between separate modules, meaning that each module has to create a completely specified interface representation to pass on to the successor module. Crucially, this means that decisions of the same type all have to be made at the same point, and cannot be postponed. Representations all have to reach the same degree of specificity at the same stage in the generation process. This is often not a good idea, e.g. in generation in machine translation: if a piece of information required by the target language is not provided locally by the source language, then instead of forcing a local decision, it would make more sense to postpone the decision between the possible alternatives until either it can be made on the basis of additional, nonlocal information¹⁰, or it is clear that the required information isn’t available at later stages either.

5.3 Underspecifiable representation spaces

Figure 2 contains a part of the generation space of Figure 1 — the semantic representation level (in the shaded box). The sentences beneath it indicate approximately the content of the semantic representations (roughly at the level of specificity of Lingo/ERG MRS — lexicalisation will at a later stage map “12am” to *noon*, *noontime*, *midday* among others, and the grammar will permit e.g. *at 12am it departs* as well as *it departs at 12am*).

If the semantic representation (SR) formalism only has one level of specificity (fully specified) then these semantic representations are all there is. However, underspecifiable SR formalisms can represent *sets* of fully specified SRs

¹⁰Example: when translating pronouns into a language which distinguishes gender in non-person nouns, the decision between the target language genders cannot usually be made at the sentence level.

(rather than individual ones) in a single representation, and then in principle each node can be explicitly represented by an (underspecified) semantic representation. One way of looking at representations that denote sets of representations is in terms of the decision(s) between the alternatives not having been made yet. That's the idea in the tree structure above the shaded box in Figure 2: the nodes and branches form an underspecifiable representation space where each node corresponds to an explicit semantic representation that is underspecified — “undecided” — between the (fully specified) representations in the leaves it dominates. The advantage of such nonspecific or undecided representations is that they encode at once information about the type of decision that hasn't been made and the available range of possible alternatives.

Making semantic representation underspecifiable can be seen as pushing back from the level of semantic representation into semantic generation (possibly even conceptual generation), encoding in the very representations information about any decisions that haven't yet been made.

With appropriately defined representations, decision space and representational space can overlap or even be identical: it is in principle possible to explicitly represent the generation space underlying any generation system as an underspecifiable representation space.

There are a number of general advantages that result from making part or all of the decision tree underlying the generation space representationally explicit:

- there no longer have to be common stages in generation where the same degree of specificity has to be reached in every individual generation process
- in principle, inputs could be taken at any level of specificity
- the architecture wouldn't have to be a strict pipeline which is often found problematic (the view of the generation process outlined in this section is probably RAGS-compatible)
- there is more flexibility about when to make decisions and when to postpone them
- and most importantly, at any stage in the generation process, the set of decisions that have already been taken, and the set of decisions yet to be taken are accessible, making it possible to look ahead and back as far as desired at any point.

5.4 AMRS as modelling a generation space decision tree

The approach to designing an underspecification framework sketched in this section is based on looking at AMRS in the way described in the last section — as an underspecifiable representation space that directly and explicitly models part of the generation space. In this view, the Lingo/ERG MRS-grammar models the very last (most shallow) part of the generation space, while an AMRS grammar models an arbitrarily large (deep) part of the generation space before it.

There are then three distinct components: (i) the **object language** of fully specified representations, call it R , (ii) some encoding of the different ways in which expressions in R can be underspecified, defining a **metalanguage** M^R , and (iii) **expansion mechanism(s)**: some way of deriving the subset of R that corresponds to an arbitrary element of M^R .

Every expansion mechanism can be combined with a range of different R . For every R many different M^R can be defined. This makes it natural to base the design of the AMRS formalism on a distinction between generic and nongeneric components. It would be useful to define a framework for a given R together with generic expansion mechanisms so that different M^R can be ‘plugged in’. In this way the representational space covered by M^R can be varied.

For the COGENT nondeterministic generation system it probably wouldn't make sense to allow different R (since R is or has to be mapped simply to MRS). So, AMRS could be designed as a

1. basic framework consisting of
 - (a) a definition of a (fully specified) representation language (the object language), and
 - (b) generic expansion mechanisms
2. a nongeneric component that defines the different ways in which representations can be underspecified (metalanguage level)

The generic underspecification framework allows the definition of an arbitrary number of different AMRSs (as well as other, completely unrelated languages of underspecified representations).

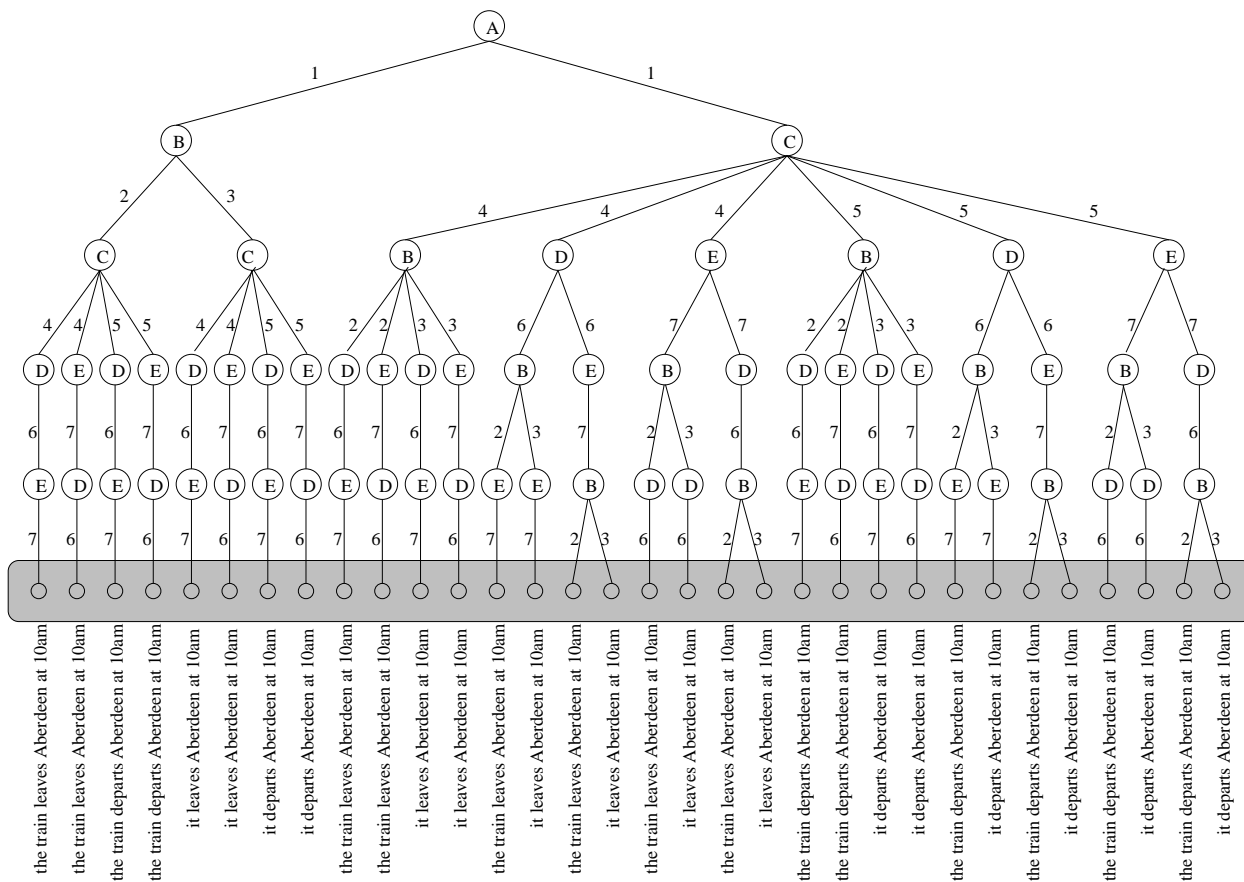


Figure 3: Underspecifiable representation space defined by CRU grammar in Example 3 .

6 Next steps: Implementing the underspecification framework and the AMRS language

6.1 Framework for representational underspecification

In COGENT, the idea in developing a representational underspecification framework for NLG is to have a tool for making the decision trees in terms of which generation spaces can be construed representationally explicit in the form of underspecification spaces. Figure 3 shows a very simple generation space (the tree diagram is analogous to that in Figure 2).

The space could be represented in the following way, where each node and the branches rooted at it are represented by the left-hand side and the right-hand side of a set of expansion rules (the letters and numbers are indices showing the corresponding nodes in Figure 2).

Example 1

```

A 1 DEPARTING_EVENT --> TRAIN DEPART
B 2 TRAIN --> the train
B 3 TRAIN --> it
C 4 DEPART --> leaves LOCATION TIME
C 5 DEPART --> departs LOCATION TIME
D 6 LOCATION --> Aberdeen
E 7 TIME --> at 10am

```

This set of expansion rules can be interpreted as a context-free grammar, and given certain constraints, the entire underspecification framework could be based on context-free techniques.

Each decision node in the tree corresponds to a nonterminal in the grammar (indexed A–E above), each branch corresponds to a single rule (numbered 1–7). The grammar generates four different fully specified strings, and 24 different underspecified strings. However, it licenses 32 different generation processes (32 different complete CFG derivations, or $S \xrightarrow[G]{*} \alpha, \alpha \in L(G)$). That's because the grammar allows the order in which decisions are taken

(nonterminals expanded) to be fairly free. Not completely: decision (A) has to be taken before any others, and (C) has to be taken before (D) and (E).

The idea is that each nonterminal together with its set of expansion rules encodes a minimal dimension of variation. The LHS can be considered a name for the variation or decision to be taken and the RHSs are the corresponding variants. From the point of view of underspecification — the expression on the LHS is underspecified between all its RHSs.

6.2 AMRS Language

The previous section outlined the beginnings of a representational framework which can be used for modelling underspecification spaces. In order to model a specific space, e.g. for the COGENT generation environment, a range of decisions have to be made. For AMRS, we need to decide what the object language is going to be. What part of the generation space is to be covered? What different types of linguistic variation is to be covered? And what external resources are going to be used to create the actual expansion rule sets?

References

- [Cahill et al., 2001] Cahill, L., Carroll, J., Evans, R., Paiva, D., Power, R., Scott, D., and van Deemter, K. (2001). From rags to riches: Exploiting the potential of a flexible generation architecture. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics and the 10th Conference of the European Chapter of the Association for Computational Linguistics (ACL-EACL '01)*, pages 98–105.
- [Carroll et al., 1999] Carroll, J., Copestake, A., Flickinger, D., and Poznanski, V. (1999). An efficient chart generator for (semi-)lexicalist grammars. In *Proceedings of the 7th European Workshop on Natural Language Generation (EWNLG '99)*, pages 86–95.
- [Copestake et al., 1999] Copestake, A., Flickinger, D., and Sag, I. (1999). Minimal recursion semantics: An introduction. Draft, available online at <http://www-csli.stanford.edu/aac/papers/newmrs.ps>.
- [Copestake et al., 2003] Copestake, A., Flickinger, D., and Sag, I. (2003). Minimal recursion semantics: An introduction. 2003 version of the paper, not publicly available.
- [Reiter, 1994] Reiter, E. (1994). Has a consensus NL generation architecture appeared and is it psycholinguistically plausible? In *Proceedings of INLG 1994*, pages 163–170.
- [Reiter and Dale, 1997] Reiter, E. and Dale, R. (1997). Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87.