An Algorithm For Generating Referential Descriptions With Flexible Interfaces

Helmut Horacek
Universität des Saarlandes
FB 14 Informatik
D-66041 Saarbrücken, Deutschland
horacek@cs.uni-sb.de

Abstract

Most algorithms dedicated to the generation of referential descriptions widely suffer from a fundamental problem: they make too strong assumptions about adjacent processing components, resulting in a limited coordination with their perceptive and linguistics data, that is, the provider for object descriptors and the lexical expression by which the chosen descriptors is ultimately realized. Motivated by this deficit, we present a new algorithm that (1) allows for a widely unconstrained, incremental, and goal-driven selection of descriptors, (2) integrates linguistic constraints to ensure the expressibility of the chosen descriptors, and (3) provides means to control the appearance of the created referring expression. Hence, the main achievement of our approach lies in providing a core algorithm that makes few assumptions about other processing components and improves the flow of control between modules.

1 Introduction

Generating referential descriptions[1] requires selecting a set of descriptors according to criteria which reflect humans preferences and verbalizing these descriptors while meeting natural language constraints. Over the last decade, (Dale, 1989, Dale, Haddock, 1991, Reiter, 1990b, Dale, Reiter, 1995), and others[2] have contributed to this issue (see the systems NAOS (Novak, 1988), EPICURE (Dale, 1988), FN (Reiter, 1990a), and IDAS (Reiter, Dale, 1992)). Nevertheless, these approaches still suffer from some crucial deficits, including limited coverage (see (Horacek, 1995, Horacek, 1996) for an improved algorithm), and too strong assumptions about adjacent processing components, namely:

- the instant availability of all descriptors for an object to be described,
- the adequate expressibility of a chosen set of descriptors in terms of lexical items.

Motivated by the resulting deficits, we develop a new algorithm that does not rely on these assumptions. It (1) allows for a widely unconstrained, incremental, and goal-driven selection of descriptors, (2) integrates linguistic constraints to ensure the expressibility of the chosen descriptors, and (3) provides means to control the appearance of the created referring expression.

This paper is organized as follows. After having introduced some basic terminology, we elaborate interface deficits of existing algorithms, form which we derive desiderata for an improved algorithm. Then we describe concepts to meet these desiderata, and we illustrate their operationalization in a schematic and in a detailed version. Finally, we demonstrate the increased functionality of the new algorithm, and we evaluate the achievements.

2 Terminology Used

In the scope of this paper, we adopt the terminology originally formulated in (Dale, 1988) and also used by several successor approaches. The referring expression to generate is required to be a *distinguishing description*, that is a description of the entity being referred to, but not to any other object in the current *context set*. A *context set* is defined as the set of entities the addressee is currently assumed to be attending to – this is similar to the set of entities in the focus spaces of the discourse focus stack in Grosz and Sidner's theory of discourse structure (Grosz, Sidner, 1986). Moreover, the *contrast set* (or, the set of *potential distractors* (McDonald, 1981)), is defined to entail all elements of the *context set* except the *intended*

---

[1] The term 'referential description' is due to Donellan (Donellan, 1966). This notion signifies a referring expression that serves the purpose of letting the hearer identify a particular object out of a set of objects assumed to be in the current focus of attention.

[2] The approach undertaken by Appelt and Kronfeld (Appelt, 1985a, Appelt, 1985b, Kronfeld, 1986, Appelt, Kronfeld, 1987) is very elaborate but it suffers from very limited coverage, missing assessments of the relative benefit of alternatives, and notorious inefficiency.

referent. In the scope of some *context set*, an attribute or a relation applicable to the *intended referent* can be assigned its *discriminatory power*,[3] that is a measure similar to the number of *potential distractors* that can be removed from the *contrast set* with confidence, because this attribute or relation does not apply to them.

3 Previous Algorithms and Deficits

The existing algorithms attempt to identify the intended referent by determining a set of descriptors attributed to that referent or to another entity related to it, thereby keeping the set of descriptors as small as possible. This minimization issue can be interpreted in different degrees of specificity, which also has consequences on the associated computational complexity. *Full brevity*, the strongest interpretation, is underlying Dale's algorithm (Dale, 1989), which produces a description entailing the minimal number of attributes possible, at the price of suffering NP-hard complexity. Two other interpretations, the *Greedy heuristic interpretation* (Dale, 1989) and the *local brevity interpretation* (Reiter, 1990a) lead to algorithms that have polynomial complexity in the same order of magnitude. The weakest interpretation, *the incremental algorithm interpretation* (Reiter, Dale, 1992), has still polynomial complexity but, unlike the last two interpretations, it is independent of the number of attributes available for building a description. Applying this interpretation may lead to the inclusion of globally redundant attributes in the final description, but this is justified by various results of psychological experiments (see the summary in (Levelt 1989)). Because of these reasons, the incremental algorithm interpretation is generally considered best now, and we adopt it for our algorithm, too.

In the realization described in (Reiter, Dale, 1992), attributes are incrementally selected according to an a priori computed domain-dependent preference list, provided each attribute contributes to the exclusion of at least one potential distractor. However, there still remains the problem of meaningfully applying this criterion in the context of nested descriptions, when the intended referent is to be described not only by attributes such as color and shape, but also in terms of other referents related to it. Neither the psychological experiments nor the realization in (Reiter, Dale, 1992) can deal with this sort of recursion. In the generalization introduced in (Horacek, 1996), descriptors of the referents are incrementally selected according to domain-dependent preference lists in a limited depth-first fashion, which leads to some sort of inflexibility through restricting the set of locally applicable descriptors. Besides, the preference list needs to be fully instantiated for each referent to be described, which constitutes a significant overhead.

An even more crucial problem lies in the fact that practically all algorithms proposed so far contend themselves with producing a set of descriptors rather than natural language expressions. They more or less implicitly assume that the set of descriptors represented as one- and two-place predicates can be expressed adequately in natural language terms. A few drastic examples should be sufficient to illustrate some of the problems that might occur due to ignoring these issues:

(a) the bottle which is on a table on which there is a cup which is besides the bottle, ...
(a problem of organization)

(b) the large, red, speedy, comfortable, ..., car
(a problem of complexity)

(c) the cup which is besides a bottle which is on a table which is left to another table and which is empty
(a scoping problem, in addition)

Altogether, two strong assumptions influence existing algorithms, namely the instant availability of all descriptors of a referent and the satisfactory expressibility of the chosen set of descriptors. They are responsible for three serious deficits negatively influencing the quality of the expression (the first one primarily causing inefficiency):

1. Applicable processing strategies are restricted because all descriptors of some referent need to be evaluated before descriptors of other referents can be considered.
2. The linguistic aspects are largely simplified and even neglected in parts. Because of the 'generation gap' (Meteer, 1992), there is no guarantee that the set of descriptors chosen can be expressed at all in the target language, not to say adequately.
3. There is no control to assess the adequacy of a certain description, for instance, in terms of structural complexity, and no feedback from linguistic form production to property selection is provided.

The first deficit restricts feasible architectures of a generation system in which such an algorithm can reasonably be embedded because flexibility and incrementality of the descriptor selection task are limited. Moreover, the underlying assumption is unrealistic in cognitive as well as in technical terms. From the perspective of human behavior, it would simply be unnecessary to determine all descriptors of a referent to be described beforehand without even attempting to generate a description; usually, just a few descriptors are sufficient for this purpose. The same considerations apply to the machine-oriented perspective: neither for a vision system nor for a knowledge-based system is it without costs to determine all descriptors of a certain object – especially for the vision system, the computational effort may be considerable.

[3] A precise definition based on numerical values assigned to attribute-value pairs is given in (Dale, 1988).

The second deficit results from ignoring that the ultimate goal envisioned consists in producing a natural language expression that satisfies the discourse goal and not merely in choosing a set of descriptors by which this goal can in principle be achieved. In general, there is no guarantee that the set of descriptors chosen can be adequately expressed in the target language, given some repertoire of lexical operators: conceptual predicates cannot always be mapped straightforwardly onto lexemes and grammatical features so that the anticipation of their composability is limited. Even more importantly, matters of grammaticality are not taken into account at all by previous algorithms. Simple cases are not problematic, for instance, when two descriptors achieve unique identification and can be expressed by a simple noun phrase consisting of a head noun and an adjective. In more complex cases, however, considerations of grammaticality such as overloading and even interference due to scoping ambiguity may become a serious concern.

The third deficit concerns the lack of control that these algorithms suffer from when assessing the structural complexity of a certain description is required, which certainly influences its communicative adequacy, too. The lack of control over the appearance of the expression to be generated is further augmented by the fact that any kind of feed-back is missing that puts the property selection facility in a position to take the needs of ultimately building a referring expression into account. Particular difficulties can be expected when a referential description needs to be produced in an incremental style, that is, portions of a surface expression are built and uttered once a further descriptor is selected, that is, prior to completion of the entire descriptor selection task.

4 Conception of a New Algorithm

Besides the primary goal of producing a distinguishing and cognitively adequate description of the intended referent, there are also the inherent secondary goals of verbally expressing the chosen descriptors in a natural way, and of applying a suitable processing strategy. In order to pursue these goals, we state the following desiderata:

1. The requirements on the descriptor providing component should widely be unconstrained, allowing for incremental and goal-driven processing.
2. A component that takes care of the expressibility of conceptual descriptors in terms of natural language expressions should be interfaced.
3. Adequate control should be provided over the complexity and components of the referring expression.

Several concepts are intended to meet these desiderata:

1. In the predecessor algorithms, attributes are taken from an a priori computed domain-dependent preference list in the indicated order, provided each attribute contributes to the exclusion of at least one potential distractor. Instead, we simply allow the responsible component to produce descriptors incrementally, even from varying referents, provided the selected descriptor is directly related to some referent already included in the expression built so far. While the precise form of this restriction is technically motivated – it guarantees that a description built this way is always connected – we believe that it is also cognitively plausible. In order to pursue the identification goal, the perception facilities preferably look for salient places in the vicinity of the object to be identified, rather than to distant places. The pre-selection obtained this way can be based on salience, eventually combined with some measure of computational effort. By applying this strategy, a best-first behavior is achieved instead of pure breadth-first (Reiter, Dale, 1992), depth-first (Dale, Haddock, 1991), and iterative deepening (Horacek, 1995, Horacek, 1996) strategies.
2. The algorithm interfaces a subprocess that incrementally attempts to build natural language expressions out of the descriptors selected. Through taking grammatical and lexical constraints into account, this process is capable of exposing expressibility problems early: expressing a proposed descriptor may require refilling an already filled slot, or integrating the mapping result of a newly inserted descriptor may lead to a global conflict such as unintended scope relations. A goal-driven aspect is added by encouraging the selection of descriptors whose images are candidates of filling empty slots in the expression built so far.
3. The algorithm enables one to control the processing aspect of building the referential description and its complexity. A parameter is provided to specify the appearance of that expression in terms of slots that are allowed to be filled. In an incremental style, where parts of the referential description are uttered prior to its completion, the slots that can be filled by the descriptor selected are substantially influenced by precedence relations (in the ordinary compositional style, this is simply identical to the set of yet empty slots).

5 Operationalization in the Algorithm

The new algorithm designed to incorporate these concepts is built on the basis of some predecessor algorithms (Dale, Haddock, 1991, Reiter, Dale 1992, Horacek, 1995, Horacek, 1996), from which we also adopt the notation. The algorithm is shown in two different degrees of precision. An informal, schematic view in Figure 1 that abstracts from technical details is complemented by a detailed pseudo-code version in Figure 2. In both versions, the lines are marked, by [S#] in the schematic view and by [C#] in the pseudo-code version to ease references from

1	*Check Success*	[S1]
	if <the intended referent is identified uniquely>	[S2]
	then <exit with an identifying description>	[S3]
	if <the complexity limit of the expression is reached>	[S4]
	then <exit with a non-identifying description>	[S5]
2	*Choose property*	[S6]
	if <no further descriptors are available>	[S7]
	then <exit with a non-identifying description>	[S8]
	else <call the descriptor selection component to propose the next property>	[S9]
	if <the descriptor does not reduce the set of potential distactors> or	[S10]
	<the referent further described is already identified uniquely> or	[S11]
	<the descriptor is inferable from the description generated so far> or	[S12]
	<the descriptor cannot be lexicalized with the given linguistic resources> or	[S13]
	<lexicalizing the descriptor would cause a scoping problem>	[S14]
	then <reject the proposed property> and goto 2	[S15]
3	*Extend description*	[S16]
	<update the linguistic resources used>	[S17]
	<determine properties which, when being lexicalized, are likely to fill yet empty slots>	[S18]
	<update the constraints holding between referents and partial descriptions>	[S19]
	goto 1	[S20]

Figure 1: Schematic presentation of the algorithm, as an abstraction from the detailed pseudo-code in Figure 2

the text. In addition, the identifiers used in the pseudo-code version are explained in Table 1 (the variables) and in Table 2 (the functions).

We first illustrate the basic structure of the procedure from some sort of a bird's eyes view. The algorithm consists of three major parts: *Check success* [S1], *Choose property* [S6], and *Extend description* [S16]; this organization stems from (Dale, Haddock, 1991) and is extended here. Basically, these parts are evaluated in sequence, which is repeated iteratively [S20]. The first part merely comprises two of the algorithm's termination criteria: [S2], which constitutes the successful accomplishment of the whole task, and [S4], which reports the failure to do this within the given limits of the linguistic resources, and corresponding return statements [S3] and [S5]. [S4] and [S5] constitute an extension to previous approaches. The second part entails a call to an external descriptor selection component [S9]. In the unlikely case that no further descriptors are available [S7] the algorithm terminates without complete success [S8]. Various tests check the suitability of the descriptor proposed in the global context: the descriptor does not contribute further to the identification task (it must be an attribute) [S10], the need of further elaborating the description of that referent to which the proposed descriptor adds information [S11], the descriptor's effective contribution to the identification task, which may be nullified due to contextual effects [S12], unavailability of lexical material to express the proposed descriptor as an extension to the referring expression composed so far [S13], and scoping problems in the attempt in extending the referring expression composed so far [S14]. The last two criteria are additions introduced in the new algorithm. In the third part, some sort of book keeping is carried out: evidence about the used lexical resources is updated [S17], descriptors that are likely to be expressible by yet empty slots are determined [S18], and relations between the context sets of all referents considered and partial descriptions are maintained [S19].

After this overview, we explain the algorithm in detail. We describe the data structure that helps controlling to whether or not a referent is identified and which the potential distractors are. Next, we illustrate the interfaces to the two major external modules. We conclude this presentation by explaining the pseudo code, thereby pointing to the corresponding parts in the schematic overview. In companion with the variables and functions explained in separated tables, this description should enable the reader to understand the functionality of the algorithm.

Throughout processing, the algorithm maintains a constraint network N which is a pair relating (a) a set of constraints, which correspond to predications over variables (properties abstracted from the individuals they apply to) to (b) sets of variables each of which fulfill these constraints in view of a given knowledge base (the context sets). The notation $N \oplus p$ is used to signify the result of adding the constraint p to the network N. In

Variable	**Description**
r, gr,v, gv	local (*r*) and global referents (*gr*) and variables (*v* and *gv*) associated with them
R	a specification of slots which the target referring expression may entail
c	(contextually-motivated) expected category of the intended referent
N	constraint network, a pair relating a set of constraints to sets of variables fulfilled by them
C	context set, indexed by variables associated with referents (e.g., C_v, C_{gv})
L	list of attribute-value pairs which corresponds to the constraint part of *N*
FD	functional description that is an appropriate lexical description expressing *L*
DD	distinguishing description, appearing as a pair <*L,FD*>
List	communicative goals to pursue, expressed by Describe(*r,v*)
<*p,r*>	property *p* ascribed to referent *r*
refs	referents already processed
P-props	properties whose images on the lexical level are likely to fill empty slots in *FD*
excluded	property-referent combinations that cannot be verbalized in the given context

Table 1: Variables used in the algorithm

addition, the notation [*r\v*]*p* is used to signify the result of replacing every occurrence of the constant *r* in *p* by variable *v* (for an algorithm to maintain consistency see AC-3 (Mackworth, 1977), as used in (Haddock, 1991)).

According to our desiderata, the new algorithm interfaces two major external modules whose precise functionality is outside the scope of this paper: *Next-Property* and *Insert-Unify*. *Next-Property* [C19], [S9] selects a cognitively-motivated candidate property to be included next. Generally applicable psychological preferences, such as basic level categories, as well as special criteria, such as degrees of applicability of local relations [Gapp 1995], may guide this selection. It is additionally influenced by two parameters: *refs*, which specifies those referent which must be directly related to the chosen descriptor, and *P-props*, which entails a list of properties whose lexical images are likely to fill yet empty slots.

Insert-Unify updates the data structure *FD* by incrementally inserting mappings of selected descriptors [C43], [S13], unless *Check-Scope* detects a global problem [C44], [S14]. This language-dependent procedure analyzes the functional description created so far for potential misinterpretations and scope ambiguities, which may occur in connection with nested postnominal modifiers or relative clauses that depend on an NP with a postnominal modifier. Examining these structures is much less expensive than a global anticipation-feedback loop, but it requires specialized grammatical knowledge. Whether the intended reading is also the preferred one depends on selectional restrictions, preference criteria, and morphological features.

Function	**Description**
Next-Property(*refs, ps*)	selects a property, influenced by the connection to referents *refs* and by properties *ps*
A(*p*)	functor to provide access to the predicate of predication *p*
find-best-value(A(*p*),*V*)	procedure to determine the value of property *p* that describes *r* according to (Dale, Reiter, 1992)
basic-level-value(*r*,A(*p*))	yields the basic level value of property *p* for referent *r*
rules-out(<A(*p*),*V*>)	yields the set of referents that are ruled out as distractors due to the value *V* of property A(*p*)
Assoc-var(*r*)	function to get access to the variable associated with referent *r*
Prototypical(*p,r*)	yields true if property *p* is prototypical for referent *r* and false otherwise
Descriptors(*r*)	yields the set of predicates entailed in *N* and holding for referent *r*
Map-to(Empty-Slots(*FD*))	yields properties which map onto the set of uninstantiated slots in *FD*
Insert-Unify(*FD*,<*v*,*p*>)	inserts a lexical description of property *p* of the referent associated with variable *v* into *FD*
Check-Scope(*FD*)	yields true if no scope problems are expected to occur and false otherwise
Slots-of(mappings(*p*))	yields the slots of the set of lexical items by which predicate *p* can be expressed
Rel(A(*p*))	yields true if descriptor *p* is a relation and false otherwise
Salient(A(*p*))	yields true if salience is assigned to property *p* and false otherwise

Table 2: Functions used in the algorithm

```
Describe (r,v,N,R,c)
DD ← nil, FD ← nil                                              [C1]
unique ← false                                                  [C2]
gr ← r, gv ← v                                                  [C3]
excluded ← nil, P-props ← nil                                   [C4]
refs ← {r}                                                      [C5]
C_v ← C_v ∩ {x | c(x)}                                          [C6]
List ← [Describe(r,v)]                                          [C7]
1 Check Success                                                 [C8]
if |C_gv| = 1 then                                              [C9]
    unique ← true                                               [C10]
return <L,FD> (as a distinguishing description)                 [C11]
endif                                                           [C12]
if |R| = 0 then                                                 [C13]
    return <L,FD>                                               [C14]
        (as a non-distinguishing description)                   [C15]
endif                                                           [C16]
2 Choose Property                                               [C17]
repeat                                                          [C18]
    <r,p> ← Next-Property(refs,P-props)                         [C19]
    if p = nil then                                             [C20]
        return <L,FD>                                           [C21]
            (as a non-distinguishing description)               [C22]
    endif                                                       [C23]
    v ← Assoc-var(r)                                            [C24]
    if Prototypical(p,r) or                                     [C25]
        ((Slots-of(Mappings(p)) ∩ R) = ∅)                       [C26]
        then excluded ← excluded ∪ {<r,p>}                      [C27]
        elseif (p in Taxonomic-Inferences                       [C28]
            (Descriptors(v))) or (|C_v| = 1) then               [C29]
                excluded ← excluded ∪ {<r,p>}                   [C30]
        endif                                                   [C31]
    endif                                                       [C32]
    if <r,p> ε excluded then                                    [C33]
        goto 2                                                  [C34]
    endif                                                       [C35]
    V = find-best-value(A(p),                                   [C36]
            basic-level-value(r,A(p)))                          [C37]
    if not (((rules-out(<A(p),V>) ≠ nil) and (V ≠ nil))         [C38]
            or Rel(A(p))) or Salient(A(p)) then                 [C39]
        excluded ← excluded ∪ {<r,p>}                           [C40]
        goto 2                                                  [C41]
    endif                                                       [C42]
    FDH ← Insert-Unify(FD, <v,p>)                               [C43]
    if not Check-Scope(FDH) then                                [C44]
        excluded ← excluded ∪ {<r,p>}                           [C45]
        goto 2                                                  [C46]
    endif                                                       [C47]
3 Extend Description                                            [C48]
FD ← FDH                                                        [C49]
R ← R \ slots(FD)                                               [C50]
P-props ← Map-to(Empty-slots(FD))                               [C51]
p ← [r\v]p                                                      [C52]
if Rel(A(p)) then                                               [C53]
    for every other constant r' in p do                         [C54]
        if Assoc-var(r') = nil then                             [C55]
            associate r' with a new, unique variable v'         [C56]
            p ← [r'\v']p                                        [C57]
            refs ← refs ∪ {r'}                                  [C58]
            List ← Append(List, Describe(r',v'))                [C59]
        endif                                                   [C60]
    next                                                        [C61]
    else set the value of attribute p to V                      [C62]
endif                                                           [C63]
N ← N ⊕ p                                                       [C64]
goto 1                                                          [C65]
```

Figure 2: Detailed pseudo-code of the new algorithm

The first part of the algorithm, 'Check Success', comprises the algorithm's termination criteria:

1. A distinguishing description is completed [C9-C11], [S2-S4], the exit in case of full success.
2. No more descriptors are globally available [C19-C22], [S7-S8]. In the predecessor algorithms, this check is done for each referent separately.
3. All available slots are filled [C13-C14], [S4-S5] – this is a new criterion.

The second part, 'Choose Property', is dedicated to test the contextual suitability of the candidate property proposed by *Next-Property*, which may be inappropriate for one of the following reasons (criteria 3. and 5. are new ones):

1. The property can be inferred from the description generated so far, or it is prototypical for the object to be identified and may thus yield a false implicature [C25, C28], [S12].
2. The object is already identified uniquely [C29], [S11].
3. The descriptor chosen cannot be mapped onto a slot of the description generated so far [C26], [S13].
4. The descriptor is an attribute, and it does not further reduce the set of potential distractors [C38], [S10].
5. Incorporating the descriptor into the functional description created so far leads to a global conflict [C43-C44], [S14].

The third part, 'Extend Description', takes care of updating some control variables. The descriptor p is fed into N [C64] goals to describe new referents reached via the relation p are put into *List* [C54-C60], [S19], all slots filled in *FD* are eliminated in R [C50], [S17], and the yet empty slots are fed into reversed lexicalization rules to yield properties collected in *P-props* [C51], [S18].

6 Effects the Algorithm Can Handle

Space restrictions do not permit a detailed presentation of the new algorithm at work. Therefore, we have confined ourselves to a sketchy description of the algorithm's behavior in a moderately complex situation. Let us assume an environment consisting of four tables (t_1 to t_4), roughly placed in a row, as depicted in Figure 2. The communicative goal is to distinguish one of the tables uniquely from the other three, by a referring expression entailing an adjective (a prenominal modifier), a category, an attribute (a postnominal modifier), and a relative clause, at most. The situation permits building a large variety of expressions for accomplishing this purpose. Some interesting cases are:

1) achieving global rather than local goal satisfaction: If t_3 is the intended referent, and on(b_1,t_3) is the descriptor selected next, adding the category of the entities on top of t_3 (here, books) is sufficient to identify t_3 uniquely. Some predecessor algorithms, for instance (Dale, Haddock 1991), would still attempt to distinguish b_1 from b_2.

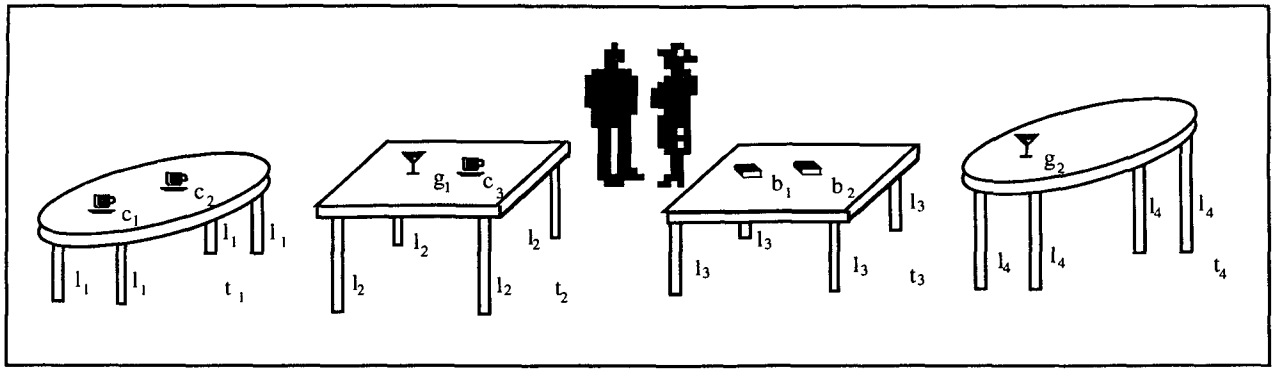

Figure 3: A scenery with tables, cups, glasses, and books

2) producing flat expressions instead of embedded ones
If t_2 is the intended referent, and on(g_1,t_2) is the descriptor selected next, another descriptor must be selected to distinguish t_2 from t_4. The descriptor selection component is free to choose on(c_3,t_2), to yield the natural, flat expression 'the table on which there are a glass and a cup'. In (Horacek 1996), the same result can be obtained through an adequate selection of search parameters. The algorithm in (Dale, Haddock 1991) would produce the less natural, embedded expression 'the table on which there is a glass besides which there is a cup' instead.

3) rejection of a descriptor because it can be inferred
If t_1 is the intended referent, and size(t_1,low) is the descriptor selected this time, another descriptor must be added, since t_3 is also subsumed by this description. If part-of(t_1,l_1) is chosen for that purpose (l_1 being the legs of t_1), the descriptor size(l_1,short) to describe l_1 further is rejected because it can be inferred from {size(t_1,low),part-of(t_1,l_1)}.

4) Rejection of a descriptor because of a clash
Let t_2 be the intended referent, and the descriptors left-of(t_3,t_2) and type(t_3,table) expressed by 'the one which is to the left of a table'. If on(g_1,t_2) is selected next, the only way to link it to the partial expression generated so far is via a relative clause, but this slot is already filled.

5) Rejection of a descriptor because of a scope problem
However, if the local relation in the previous example is expressed by 'the one to the left of a table', adding a relative clause expressing the objects on t_3 would still work badly because the addressee would interpret these objects to be placed on t_2 – *Check-Scope* should recognize this reference problem.

7 Evaluating the Algorithm

The examples discussed in the previous section demonstrate that our procedure avoids many of the deficits previous algorithms suffer from. Therefore, it provides excellent prerequisites for producing natural referring expressions in terms of both, descriptors selected and structural appearance. Whether this is actually the case depends primarily on the quality of the external components, the descriptor selection and the lexicalization component and, to some minor extent, on the parameterization of the structural appearance of the referring expression to be produced.

As far as its complexity is concerned, the algorithm is in some sense even more efficient than its predecessors, because it does not require complete lists of descriptors to be produced for each referent. However, this saving is partially nullified by the additional operations incorporated, especially by the application of lexicalization operators and scoping verifications. Nevertheless, an overall analysis of the algorithm's complexity is hardly possible in a general sense because

- the operations in this algorithm are rather heterogeneous, and their relative costs are far from clear,
- the costs of individual operations, such as descriptor computation in the descriptor selection component and constraint network maintenance, may vary significantly in dependency of the underlying representation, especially if the primary representation is a pictorial rather than a propositional one.

8 Conclusion

In this paper, we have presented a new algorithm for generating referential descriptions which exhibits some extraordinary capabilities:

- Descriptors can be selected in a goal-driven and incremental fashion, with contributions from varying referents interleaving with one another.
- A component is interfaced which attempts to express the descriptors chosen on the lexical representation level to encounter expressibility problems.
- The structural appearance of the resulting referential description can be controlled.

Major problems for the future are an even tighter integration of the algorithm in the generation process as a whole and finding adequate concepts for dealing with negation and sets.

References

Doug Appelt. 1985a. Planning English Referring Expressions. *Artificial Intelligence,* 26:1-33.

Doug Appelt. 1985b. Some Pragmatic Issues in the Planning of Definite and Indefinite Referring Expressions. *In 23rd Annual Meeting of the Association for Computational Linguistics*, pages 198-203. Association for Computational Linguistics, Morristown, New Jersey.

Doug Appelt, and Amichai Kronfeld. 1987. A Computational Model of Referring. In Proceedings of the *10th International Joint Conference on Artificial Intelligence*, pages 640-647, Milano, Italy.

Robert Dale. 1988. Generating Referring Expressions in a Domain of Objects and Processes. PhD Thesis, Centre for Cognitive Science, University of Edinburgh.

Robert Dale. 1989. Cooking Up Referring Expressions. In *27th Annual Meeting of the Association for Computational Linguistics*, pages 68-75, Vancouver, Canada. Association for Computational Linguistics, Morristown, New Jersey.

Robert Dale, and Nick Haddock. 1991. Generating Referring Expressions Involving Relations. In Proceedings of the *European Chapter of the Association for Computational Linguistics*, pages 161-166, Berlin, Germany.

Robert Dale, and Ehud Reiter. 1995. Computational Interpretations of the Gricean Maxims in the Generation of Referring Expressions. *Cognitive Science*, 19:233-263.

K. Donellan. 1966. Reference and Definite Description. *Philosophical Review*, 75:281-304.

Klaus-Peter Gapp. 1995. Efficient Processing of Spatial Relations in General Object Localization Tasks. In Proceedings of the *Eighth Australian Joint Conference on Artificial Intelligence*, Canberra, Australia.

Barbara Grosz, and Candace Sidner. 1986. Attention, Intention, and the Structure of Discourse. *Computational Linguistics*, 12:175-206.

Nicolas Haddock. 1991. Linear-Time Reference Evaluation. Technical Report, Hewlett Packard Laboratories, Bristol.

Helmut Horacek. 1995. More on Generating Referring Expressions. In Proceedings of the *5th European Workshop on Natural Language Generation*, pages 43-58, Leiden, The Netherlands.

Helmut Horacek. 1996. A New Algorithm for Generating Referring Expressions. In Proceedings of the *8th European Conference on Artificial Intelligence*, pages 577-581, Budapest, Hungary.

Amichai Kronfeld. 1986. Donellan's Distinction and a Computational Model of Reference. In *24th Annual Meeting of the Association for Computational Linguistics*, pages 186-191. Association for Computational Linguistics, Morristown, New Jersey.

William Levelt. 1989. *Speaking: From Intention to Articulation.* MIT Press.

Alan Mackworth. 1977. Consistency in Networks of Relations. *Artificial Intelligence*, 8:99-118.

David McDonald. 1981. Natural Language Generation as a Process of Decision Making Under Constraints. PhD Thesis, MIT, Cambridge, Massachusetts.

Marie Meteer. 1992. Expressibility and the Problem of Efficient Text Planning. Pinter Publishers, London.

Hans-Joachim Novak. 1988. Generating Referring Phrases in a Dynamic Environment. In M. Zock, G. Sabah, editors, *Advances in Natural Language Generation*, Vol. 2, pages 76-85, Pinter publishers, London.

Ehud Reiter. 1990a. *The Computational Complexity of Avoiding Conversational Implicatures.* In *28th Annual Meeting of the Association for Computational Linguistics*, pages 97-104, Pittsburgh, Pennsylvania. Association for Computational Linguistics, Morristown, New Jersey.

Ehud Reiter. 1990b. Generating Descriptions that Exploit a User's Domain Knowledge. In R. Dale, C. Mellish, M. Zock, editors, *Current Issues in Natural Language Generation,* pages 257-285, Academic Press, New York.

Ehud Reiter, and Robert Dale. 1992. Generating Definite NP Referring Expressions. In Proceedings of the *International Conference on Computational Linguistics,* Nantes, France.