TEXTUAL ECONOMY THROUGH CLOSE COUPLING OF SYNTAX AND SEMANTICS

Matthew Stone Bonnie Webber
Dept. of Computer & Information Science
University of Pennsylvania
200 South 33rd Street
Philadelphia PA 19104-6389 USA
matthew@linc.cis.upenn.edu, bonnie@central.cis.upenn.edu

Abstract

We focus on the production of efficient descriptions of objects, actions and events. We define a type of efficiency, *textual economy*, that exploits the hearer's recognition of inferential links to material elsewhere within a sentence. Textual economy leads to efficient descriptions because the material that supports such inferences has been included to satisfy independent communicative goals, and is therefore *overloaded* in the sense of Pollack [18]. We argue that achieving textual economy imposes strong requirements on the representation and reasoning used in generating sentences. The *representation* must support the generator's simultaneous consideration of syntax and semantics. *Reasoning* must enable the generator to assess quickly and reliably at any stage how the hearer will interpret the current sentence, with its (incomplete) syntax and semantics. We show that these representational and reasoning requirements are met in the SPUD system for sentence planning and realization.

1 Introduction

The problem we address is that of producing efficient descriptions of objects, collections, actions, events, etc. (i.e., any *generalized individual* from a rich ontology for Natural Language such as those described in [2] and advocated in [9]). We are interested in a particular kind of efficiency that we call *textual economy*, which presupposes a view of sentence generation as *goal-directed activity* that has broad support in Natural Language Generation (NLG) research [1, 5, 15, 17]. According to this view, a system has certain communicative intentions that it aims to fulfill in producing a description. For example, the system might have the goal of identifying an individual or action α to the hearer, or ensuring that the hearer knows that α has property *P*. Such goals can be satisfied explicitly by assembling appropriate syntactic constituents—for example, satisfying the goal of identifying an individual using a noun phrase that refers to it or identifying an action using a verb phrase that specifies it. *Textual economy* refers to satisfying such goals implicitly, by exploiting the hearer's (or reader's) recognition of inferential links to material elsewhere in the sentence that is there to satisfy independent communicative goals. Such material is therefore *overloaded* in the sense of [18].[1] While there are other ways of increasing the efficiency of descriptions (Section 5), our focus is on the efficiency to be gained by viewing a large part of generation in terms of describing (generalized) individuals.

Achieving this however places strong requirements on the representation and reasoning used in generating sentences. The *representation* must support the generator's proceeding incrementally through the syntax and semantics of the sentence as a whole. The *reasoning* used must enable the generator to assess quickly and reliably at any stage how the hearer will interpret the current sentence, with its (incomplete) syntax and semantics. Only by evaluating the status of such key questions as

- what (generalized) individuals could the sentence (or its parts) refer to?

[1] Pollack used the term *overloading* to refer to cases where a single intention to act is used to wholly or partially satisfy several of an agent's goals simultaneously.

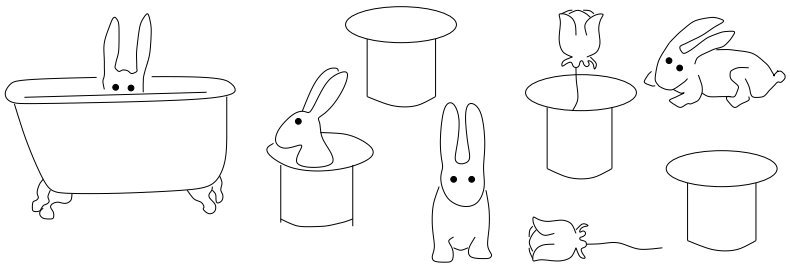


Figure 1: "Remove the rabbit from the hat."

- what (generalized) individuals would the hearer take the sentence to refer to?
- what would the sentence invite the hearer to conclude about those individuals?
- how can this sentence be modified or extended?

can the generator recognize and exploit an opportunity for textual economy.

These representational and reasoning requirements are met in the SPUD system for sentence planning and realization [26, 27]. SPUD draws on earlier work by Appelt [1] in building sentences using planning techniques. SPUD plans the syntax and semantics of a sentence by incorporating lexico-grammatical entries into a partial sentence one-by-one and incrementally assessing the answers to the questions given above. In this paper, we describe the intermediate representations that allow SPUD to do so, since these representations have been glossed over in earlier presentations [26, 27]. Reasoning in SPUD is performed using a fast modal theorem prover [24, 25] to keep track both of what the sentence *entails* and what the sentence *requires* in context. By reasoning about the *predicated* relationships *within* clauses and the *informational* relationships [16] *between* clauses, SPUD is able to generate sentences that exhibit two forms of textual economy: *referential interdependency* among noun phrases within a single clause, and *pragmatic overloading* of clauses in instructions [7].

For an informal example of the textual economy to be gained by taking advantage of *predicated* relationships within clauses, consider the scene pictured in Figure 1 and the goal of getting the hearer to take the rabbit currently in the hat out of the hat it's currently in. Even though there are several rabbits, several hats, and even a rabbit in a bathtub and a flower in a hat, it would be sufficient here to issue the command:

(1) *Remove the rabbit from the hat.*

It suffices because one of the semantic features of the verb *remove*—that its object (here, the rabbit) starts out in the source (here, the hat)—distinguishes the intended rabbit and hat in Figure 1 from the other ones.

Pragmatic overloading [7] illustrates how an informational relation between clauses can support textual economy in the clauses that serve as its "arguments". In [7], we focused on describing (complex) actions, showing how a clause interpreted as conveying the *goal* β or *termination condition* τ of an action α partially specified in a related clause forms the basis of a constrained inference that provides additional information about α. For example,

(2a) *Hold the cup under the spigot...*

(2b) *...to fill it with coffee.*

Here, the two clauses (2a) and (2b) are related by purpose—specifically, enablement. The action α described in (2a) will enable the actor to achieve the goal β described in (2b). While α itself does not specify the orientation of the cup under the spigot, its purpose β can lead the hearer to an appropriate choice—to fill a cup with coffee, the cup must be held vertically, with its concavity pointing upwards. As noted in [7], this constraint depends crucially on the *purpose* for which α is performed. The purpose specified in (3b) does not constrain cup orientation in the same way:

(3a) *Hold the cup under the faucet...*

(3b) *...to wash it.*

Examples like (1) and (2) suggest that the natural locality for sentence planning is in a description of a generalized individual. Even though such descriptions may play out over several clauses (or even sentences), the predications within clauses and the informational relations across clauses of a description give rise to similar textual economies, that merit a similar treatment.

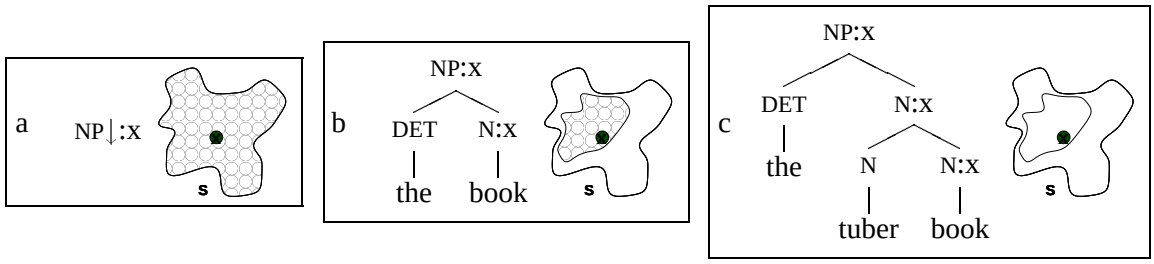
2 SPUD

An NLG system must satisfy at least three constraints in mapping the content planned for a sentence onto the string of words that realize it [4, 13, 20]. Any fact to be communicated must be fit into an abstract grammatical structure, including lexical items. Any reference to a domain entity must be elaborated into a description that distinguishes the entity from its *distractors*—the salient alternatives to it in context. Finally, a surface form must be found for this conceptual material.

In one architecture for NLG systems that is becoming something of a standard [22], these tasks are performed in separate stages. For example, to refer to a uniquely identifiable entity *x* from the common ground, first a set of concepts is identified that together single out *x* from its distractors in context. Only later is the syntactic structure that realizes those concepts derived.

SPUD [26, 27] integrates these processes in generating a description—producing both syntax and semantics simultaneously, in stages, as illustrated in (4).

(4)



Each step adds to the representation a lexicalized entry encoded as an elementary tree in Feature-based Lexicalized Tree-Adjoining Grammar (LTAG) [23]. A tree may contain multiple lexical items (cf. (4)b). Each such tree is paired with logical formulae that, by referring to a rich discourse model, characterize the semantic and pragmatic contribution that it makes to the sentence. We give a detailed example of SPUD's processing in Section 3 and describe in Section 4 the reasoning methods we use to derive computational representations like the set of distractors shown in (4). For now, a general understanding of SPUD suffices—this is provided by the summary in Figure 2.

The procedure in Figure 2 is sufficiently general so that SPUD can use similar steps to construct both definite and indefinite referring forms. The main difference lies how alternatives are evaluated. When an indefinite referring form is used to refer to a *brand-new* generalized individual [19] (an object, for example,

- Start with a tree with one node (e.g., S, NP) and one or more referential or informational goals.
- While the current tree is incomplete, or its references are ambiguous to the hearer, or its meaning does not fully convey the informational goals (provided progress is being made):
  - consider the trees that extend the current one by the addition (using LTAG operations) of a true and appropriate lexicalized descriptor;
  - rank the results based on local factors (e.g., completeness of meaning, distractors for reference, unfilled substitution sites, specificity of licensing conditions);
  - make the highest ranking new tree the current tree.

Figure 2: An outline of the SPUD algorithm

or an action in an instruction), the object is marked as new and does not have to be distinguished from others because the hearer creates a fresh "file card" for it. However, because the domain typically provides features needed in an appropriate description for the object, SPUD continues its incremental addition of content to convey them. When an indefinite form is used to refer to an old object that cannot be distinguished from other elements of a uniquely identifiable set (typically an *inferrable* entity [19]), a process like that illustrated in (4) must build a description that identifies this set, based on the known common properties of its elements.

Several advantages of using LTAG in such an integrated system are described in [27] (See also previous work on using TAG in NLG such as [11] and [29]). These advantages include:

- *Syntactic constraints can be handled early and naturally.* In the problem illustrated in (4), SPUD directly encodes the syntactic requirement that a description should have a head noun—missing from the concept-level account—using the NP substitution site.
- *The order of adding content is flexible.* Because an LTAG derivation allows modifiers to adjoin at any step (unlike a top-down CFG derivation), there is no tension between providing what the syntax requires and going beyond what the syntax requires.
- *Grammatical knowledge is stated once only.* All operations in constructing a sentence are guided by LTAG's lexicalized grammar; by contrast, with separate processing, the lexicon is split into an inventory of concepts (used for organizing content or constructing descriptions) and a further inventory of concepts in correspondence with some syntax (for surface realization).

This paper delineates a less obvious, but equally significant advantage that follows from the ability to consider multiple goals in generating descriptions, using a representation and a reasoning process in which syntax and semantics are more closely linked:

- *It naturally supports textual economy.*

3 Achieving Textual Economy

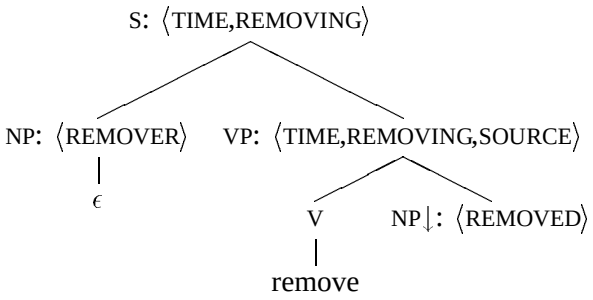
To see how SPUD supports textual economy, consider first how SPUD might derive the instruction in Example (1). For simplicity, this explanation assumes SPUD makes a nondeterministic choice from among available lexical entries; this suffices to illustrate how SPUD can realize the textual economy of this example.

A priori, SPUD has a general goal of describing a new action that the hearer is to perform, by making sure the hearer can identify the key features that allow its performance. For (1), then, SPUD is given two

features of the action to be described: it involves motion of an intended object by the agent, and its result is achieved when the object reaches a place decisively away from its starting point.

The first time through the loop of Figure 2, SPUD must expand an S node. One of the applicable moves is to substitute a lexical entry for the verb *remove*. Of the elements in the verb's LTAG tree family, the one that fits the instructional context is the imperative tree of (5).

(5a) Syntax:



(5b) Semantics: nucleus(PREP, REMOVING, RESULT) ∧ in(PREP, start(TIME), REMOVED, SOURCE) ∧
caused-motion(REMOVING, REMOVER, REMOVED) ∧
away(RESULT, end(TIME), REMOVED, SOURCE)

The tree given in (5a) specifies that *remove* syntactically *satisfies* a requirement to include an S, *requires* a further NP to be included (describing what is removed), and *allows* the possibility of an explicit VP modifier that describes what the latter has been removed from.[2] The semantics in (5b) consists of a set of features, formulated in an ontologically promiscuous semantics, as advocated in [9]. It follows [14] in viewing events as consisting of a preparatory phase, a transition, and a result state (what is called a *nucleus* in [14]). The semantics in (5b) describes all parts of a *remove* event: In the preparatory phase, the object (REMOVED) is in/on SOURCE. It undergoes motion caused by the agent (REMOVER), and ends up away from SOURCE in the result state.

Semantic features are used by SPUD in one of two ways. Some make a *semantic contribution* that specifies new information—these add to what new information the speaker can convey with the structure. Others simply impose a *semantic requirement* that a fact must be part of the conversational record—these figure in ruling out distractors.

For this instruction, SPUD treats the CAUSED-MOTION and AWAY semantic features as semantic contributions. It therefore determines that the use of this item communicates the needed features of the action. At the same time, it treats the IN feature—because it refers to the shared initial state in which the instruction will be executed—and the NUCLEUS feature—because it simply refers to our general ontology—as semantic requirements. SPUD therefore determines that the only ⟨REMOVED,SOURCE⟩ pairs that the hearer might think the instruction could refer to are pairs where REMOVED starts out in/on SOURCE as the action begins.

Thus, SPUD derives a triple effect from use of the word *remove*—increasing *syntactic satisfaction*, making *semantic contributions* and satisfying *semantic requirements*—all of which contribute to SPUD's task of completing an S syntactic constituent that conveys needed content and refers successfully. Such multiple effects make it natural for SPUD to achieve textual economy. Positive effects on any of the above dimensions can suffice to merit inclusion of an item in a given sentence. However, the effects of inclusion may go beyond this: even if an item is chosen for its semantic contribution, its semantic requirements can

[2]Other possibilities are that SOURCE is not mentioned explicitly, but is rather inferred from (1) the previous discourse or, as we will discuss later, (2) either the predicated relationships within the clause or its informational relationship to another clause.

still be exploited in establishing whether the current lexico-syntactic description is sufficient to identify an entity, and its syntactic contributions can still be exploited to add further content.

Since the current tree is incomplete and referentially ambiguous, SPUD repeats the loop of Figure 2, considering trees that extend it. One option is to adjoin at the VP the entry corresponding to *from the hat*. In this compound entry, *from* matches the verb and *the* matches the context; *hat* carries semantics, requiring that SOURCE be a hat. After adjunction, the requirements reflect both *remove* and *hat*; reference, SPUD computes, has been narrowed to the hats that have something in/on them (the rabbit, the flower).

Another option is to substitute the entry for *the rabbit* at the object NP; this imposes the requirement that REMOVED be a rabbit. Suppose SPUD discards this option in this iteration, making the other (perhaps less referentially ambiguous) choice. At the next iteration, *the rabbit* still remains an option. Now combining with *remove* and *hat*, it derives a sentence that SPUD recognizes to be complete and referentially unambiguous, and to satisfy the informational goals.

Now we consider the derivation of (2), which shows how an informational relation between clauses can support textual economy in the clauses that serve as its "arguments". SPUD starts with the goal of describing the holding action in the main clause, and (if possible) also describing the filling action and indicating the purpose relation (i.e., *enablement*) between them. For the *holding* action, SPUD's goals include making sure that the sentence communicates where the cup will be held and how it will be held (i.e., UPWARD). SPUD first selects an appropriate lexico-syntactic tree for imperative *hold*; SPUD can choose to adjoin in the purpose clause next, and then to substitute in an appropriate lexico-syntactic tree for *fill*. After this substitution, the semantic contributions of the sentence describe an action of *holding an object* which *generates* an action of *filling that object*. As shown in [7], these are the premises of an inference that the object is held upright during the filling. When SPUD queries its goals at this stage, it thus finds that it has in fact conveyed how the cup is to be held. SPUD has no reason to describe the orientation of the cup with additional content.

Additional examples of using SPUD to generate instructions can be found in [3, 25].

4 Assessing interpretation in SPUD

This section describes in a bit more detail how SPUD computes the effects of incorporating a particular lexical item into the sentence being constructed. For a more extensive discussion, see [25].

SPUD's computations depend on its representation of overall contextual background, including the status of propositions and entities in the discourse. For the purpose of generating instructions to a single hearer, we assume that any *proposition* falls either within the private knowledge of the speaker or within the common ground that speaker and hearer share. We implement this distinction by specifying facts in a modal logic with an explicit representation of knowledge: $[\mathrm{S}]p$ means that the speaker knows p; $[\mathrm{C}]p$ means that p is part of the common ground. Each *entity*, e, comes with a context set $D(e)$ including it and its distractors. Linguistically, when we have $a \in D(b)$ but not $b \in D(a)$, then a is more salient than b.

This conversational background serves as a resource for constructing and evaluating a three-part state-record for an incomplete sentence, consisting of:

- An *instantiated* tree describing the syntactic structure of the sentence under construction. Its nodes are labeled by a sequence of variables $\mathbf{v}$ indicating the patterns of coreference in the tree; but the tree also records that the speaker intends $\mathbf{v}$ to refer to a particular sequence of generalized individuals $\mathbf{e}$.
- The *semantic requirements* of the tree, represented by a formula $R(\mathbf{v})$. This formula must match facts in the common ground; in our modal specification, such a match corresponds to a proof whose conclusion instantiates $[\mathrm{C}]R(\mathbf{v})$. In particular, the speaker ensures that such a proof is available when $\mathbf{v}$ is instantiated to the entities $\mathbf{e}$ that the speaker means to refer to. This determines what alternative

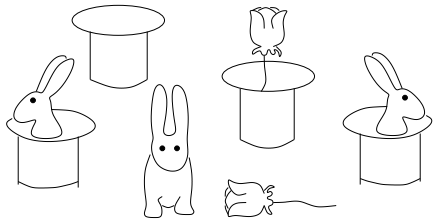
referents that the hearer may still consider: $\{ \mathbf{a} \in D(\mathbf{e}) \mid [\mathrm{C}]R(\mathbf{a}) \}$. The semantic requirements of the tree result from conjoining the requirements $R_i(\mathbf{v}_i)$ of the individual lexical items from which the state is derived.

- The *semantic contributions* of the tree, represented by a formula $N(\mathbf{v})$; again, this is the conjunction of the contributions $N_i(\mathbf{v}_i)$ of the individual items. These contributions are added to the common ground, allowing both speaker and hearer to draw shared conclusions from them. This has inspired the following test for whether a goal to communicate G has been indirectly achieved. Consider the content of the discourse as represented by $[\mathrm{C}]$, augmented by what this sentence will contribute (assuming we identify entities as needed for reference): $N(\mathbf{e})$. Then if G follows, the speaker has conveyed what is needed.

When SPUD considers extending a state by a lexical item, it must be able to update each of these records quickly. The heart of SPUD's approach is logic programming [25], which links complexity of computation and complexity of the domain in a predictable way. For example, informational goals are assessed by the query $[\mathrm{C}](N(\mathbf{e}) \supset G)$. This leaves room for inference when necessary, without swamping SPUD; in practice, G is often a primitive feature of the domain and the query reduces to a simple matching operation. Another source of tractability comes from combining logic programming with special-purpose reasoning. For example, in computing reference, $\{ \mathbf{a}_i \in D(\mathbf{e}_i) \mid [\mathrm{C}]R_i(\mathbf{a}_i) \}$ is found using logic programming but the overall set of alternatives is maintained using arc-consistency constraint-satisfaction, as in [6, 8].

SPUD must also settle which semantic features are taken to constitute the semantic *requirements* of the lexical item and which are taken to constitute its semantic *contributions*.[3] When SPUD partitions the semantic features of the lexical item, as many features as possible are cast as requirements—that is, the item links as strongly with the context as possible. In some cases, the syntactic environment may further constrain this assignment. For example, we constrain items included in a definite NP to be semantic *requirements*, while the main verb in an indicative sentence is usually taken to make a semantic *contribution*. (Exceptions to such a policy are justified in [28].)

[3]These can vary with context: consider a variant on Figure 1, where the hearer is asked "What just happened?".



One possible response — "I have removed the rabbit from the hat" — refers successfully, despite the many rabbits and hats, because there is still only one rabbit in this scene that could have been removed from a hat. Here, where the scene is taken as shared, what is taken as a semantic requirement of *remove*—that the rabbit ends up away from the hat—is used to identify a unique rabbit. This contrasts with the previous "rabbit" example where, taking the scene in Figure 1 as shared, the command "Remove the rabbit from the hat" takes as its semantic requirement that the rabbit be in the hat and uses it for unique identification. Note that if the above scene is not taken as shared, both are then taken as semantic contributions, and "I have removed a rabbit from a hat" becomes an acceptable answer.

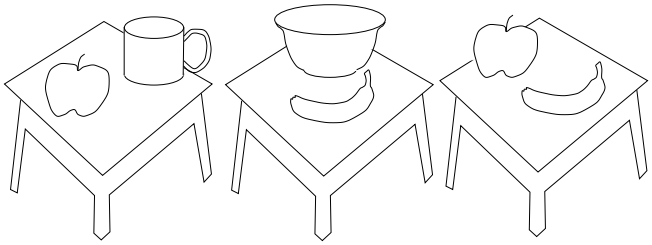


Figure 3: "The table with the apple and with the banana"

5 Other Methods that Contribute to Efficient Descriptions

This section contrasts SPUD—and its close coupling of syntax and semantics—with prior work on generating more concise descriptions by considering the effects of broader goals,[4] starting with Appelt [1]. Appelt's planning formalism includes plan-critics that can detect and collapse redundancies in sentence plans. However, his framework treats subproblems in generation as independent by default; and writing tractable and general critics is hampered by the absence of abstractions like those used in SPUD to simultaneously model the syntax and the interpretation of a whole sentence.

[6, 10, 12], in contrast, use specialized mechanisms to capture particular descriptive efficiencies. By using syntax to work on inferential and referential problems simultaneously, SPUD captures such efficiencies in a uniform procedure. For example, in [12], McDonald considers descriptions of events in domains which impose strong constraints on what information about events is semantically relevant. He shows that such material should and can be omitted, if it is both syntactically optional and inferentially derivable:

> FAIRCHILD Corporation (Chantilly VA) Donald E Miller was named senior vice president and general counsel, succeeding Dominic A Petito, who resigned in November, at this aerospace business. Mr. Miller, 43 years old, was previously principal attorney for Temkin & Miller Ltd., Providence RI.

Here, McDonald points out that one does not need to explicitly mention the position that Petito resigned from in specifying the resignation sub-event, since it must be the same as the one that Miller has been appointed to. This can be seen as a special case of pragmatic overloading.

Meanwhile, Dale and Haddock [6] consider generating interacting references, building on Haddock's work on reference resolution [8]. Their example NP, *the rabbit in the hat*, refers successfully in a context with many rabbits and many hats, so long as only one of the rabbits, r_5 say, is actually in one of the hats, h_3 say. Like (1), the efficiency of this description comes from the uniqueness of this rabbit-hat pair. However, Dale and Haddock construct NP semantics in isolation and adopt a fixed, depth-first strategy for adding content. Horacek [10], challenges this strategy with examples that show the need for modification at multiple points in an NP. For example, (6) refers with respect to the scene in Figure 3.

(6) *the table with the apple and with the banana*.

(6) identifies a unique table by exploiting its association with two objects it supports: the apple and the banana that are on it. (Note the other tables, apples and bananas in the figure—and even tables with apples and tables with bananas.) Reference to one of these—the apple, say—is incorporated into the description

[4] Other ways of making descriptions more concise, such as through the use of anaphoric and deictic pronouns (or even pointing, in multi-modal contexts), are parasitic on the hearer's focus of attention, which can (in large part) be defined independently of goal-directed features of text.

first; then that (subordinate) entity is identified by further describing the table (higher up).[5] By considering sentences rather than isolated noun phrases, SPUD extends such descriptive capacities even further.

6 Remarks and Conclusion

In this paper, we have shown how the semantics associated with predication within clauses and informational relations between clauses can be used to achieve textual economy in a system (SPUD) that closely couples syntax and semantics. In both cases, efficiency depends only on the informational consequences of current lexico-syntactic choices in describing the *generalized* individual of interest; there is no appeal to information available in the discourse context, which is already well-known as a source of economy, licensing the use of anaphoric and deictic forms, the use of ellipsis, etc. Thus, we claim that this approach truly advances current capabilities in NLG.

Finally, we must make clear that we are talking about the *possibility* of producing a particular description (one in which a wider range of inferrable material is elided); we are *not* making claims about a particular algorithm that exploits such a capability. Thus it is not relevant here to question computational complexity or look for a comparison with algorithms previously proposed by Dale, Reiter, Horacek and others [4, 10, 21] that compute "minimal" descriptions of some form. Currently, the control algorithm used in the SPUD generator is the simple greedy algorithm described in [26, 27] and summarized in Figure 2. The important point is that the process enables inferences to be performed that allow more economical texts: the next step is to address the complexity issues that these other authors have elaborated and show how SPUD's description extension and verification process can be incorporated into a more efficient or more flexible control structure.

7 Acknowledgments

Support for this work has come from an IRCS graduate fellowship, the Advanced Research Project Agency (ARPA) under grant N6600194C6-043 and the Army Research Organization (ARO) under grant DAAHO494GO426. Thanks go to Mark Steedman and members of the SPUD group: Christy Doran, Gann Bierner, Tonia Bleam, Julie Bourne, Aravind Joshi, Martha Palmer, and Anoop Sarkar.

References

[1] Douglas Appelt. *Planning English Sentences*. Cambridge University Press, Cambridge England, 1985.

[2] Emmon Bach. *Informal Lectures on Formal Semantics*. State University of New York Press, Albany, NY, 1989.

[3] Juliet Bourne. Generating effective instructions: Knowing when to stop. PhD Thesis Proposal, Department of Computer & Information Science, University of Pennsylvania, July 1998.

[4] Robert Dale. *Generating Referring Expressions in a Domain of Objects and Processes.* PhD thesis, Centre for Cognitive Science, University of Edinburgh, 1989.

[5] Robert Dale. *Generating Referring Expressions.* MIT Press, Cambridge MA, 1992.

[6] Robert Dale and Nicholas Haddock. Content determination in the generation of referring expressions. *Computational Intelligence*, 7(4):252–265, 1991.

[7] Barbara Di Eugenio and Bonnie Webber. Pragmatic overloading in natural language instructions. *Internationl Journal of Expert Systems,* 9(2):53–84, 1996.

[5] Horacek also stresses that there is no need to describe these auxiliary objects separately or identify them uniquely. For example, *the table with two fruits* provides a possible alternative to (6). We see no principled obstacle to expressing the same insight in SPUD. However, because of SPUD's close coupling between syntax and semantics, this analysis must await development of a semantic analysis of plurality in SPUD.

[8] Nicholas Haddock. *Incremental Semantics and Interactive Syntactic Processing.* PhD thesis, University of Edinburgh, 1989.

[9] Jerry R. Hobbs. Ontological promiscuity. In *Proceedings of ACL*, pages 61–69, 1985.

[10] Helmut Horacek. More on generating referring expressions. In *Proceedings of the Fifth European Workshop on Natural Language Generation*, pages 43–58, Leiden, 1995.

[11] Aravind K. Joshi. The relevance of tree adjoining grammar to generation. In Gerard Kempen, editor, *Natural Language Generation*, pages 233–252. Martinus Nijhoff Press, Dordrecht, The Netherlands, 1987.

[12] David McDonald. Type-driven suppression of redundancy in the generation of inference-rich reports. In Robert Dale, Eduard Hovy, Dietmar Rösner, and Oliviero Stock, editors, *Aspects of Automated Natural Language Generation: 6th International Workshop on Natural Language Generation*, Lecture Notes in Artificial Intelligence 587, pages 73–88. Springer Verlag, Berlin, 1992.

[13] Marie W. Meteer. Bridging the generation gap between text planning and linguistic realization. *Computational Intelligence*, 7(4):296–304, 1991.

[14] Marc Moens and Mark Steedman. Temporal ontology and temporal reference. *Computational Linguistics*, 14(2):15–28, 1988.

[15] Johanna Moore. *Participating in Explanatory Dialogues*. MIT Press, Cambridge MA, 1994.

[16] Johanna Moore and Martha Pollack. A problem for RST: The need for multi-level discouse analysis. *Computational Linguistics*, 18(4):537–544, 1992.

[17] Johanna D. Moore and Cécile L. Paris. Planning text for advisory dialogues: capturing intentional and rhetorical information. *Computational Linguistics*, 19(4):651–695, 1993.

[18] Martha Pollack. Overloading intentions for efficient practical reasoning. *Noûs*, 25:513–536, 1991.

[19] Ellen Prince. Toward a taxonomy of given-new information. In P. Cole, editor, *Radical Pragmatics*. Academic Press, 1981.

[20] Owen Rambow and Tanya Korelsky. Applied text generation. In *ANLP*, pages 40–47, 1992.

[21] Ehud Reiter. A new model of lexical choice for nouns. *Computational Intelligence*, 7(4):240–251, 1991.

[22] Ehud Reiter and Robert Dale. Building applied natural language generation systems. *Natural Language Engineering*, 3:57–88, 1997.

[23] Yves Schabes. *Mathematical and Computational Aspects of Lexicalized Grammars*. PhD thesis, Computer Science Department, University of Pennsylvania, 1990.

[24] Matthew Stone. Applying theories of communicative action in natural language generation using logic programming. In *AAAI Fall Symposium on Communicative Action*, 1997.

[25] Matthew Stone. *Modality in Dialogue: Planning, Pragmatics and Computation*. PhD thesis, University of Pennsylvania, 1998.

[26] Matthew Stone and Christine Doran. Paying heed to collocations. In *International Natural Language Generation Workshop*, pages 91–100, 1996.

[27] Matthew Stone and Christine Doran. Sentence planning as description using tree-adjoining grammar. In *Proceedings of ACL*, pages 198–205, 1997.

[28] Lyn Walker. *Informational redundancy and resource bounds in dialogue*. PhD thesis, Department of Computer & Information Science, University of Pennsylvania, 1993. Institute for Research in Cognitive Science report IRCS-93-45.

[29] Gijoo Yang, Kathleen F. McCoy, and K. Vijay-Shanker. From functional specification to syntactic structures: systemic grammar and tree-adjoining grammar. *Computational Intelligence*, 7(4):207–219, 1991.