Generating Text with a Theorem Prover

Iván I. Garibay
School of Computer Science
University of Central Florida
Orlando, FL
igaribay@cs.ucf.edu

Abstract

The process of documenting designs is tedious and often error-prone. We discuss a system that automatically generates documentation for the single step transition behavior of Statecharts with particular focus on the correctness of the result in the sense that the document will present all and only the facts corresponding to the design being documented.

Our approach is to translate the Statechart into a propositional formula, then translate this formula into a natural language report. In the later translation pragmatic effects arise due to the way the information is presented. Whereas such effects can be difficult to quantify, we account for them within an abstract framework by applying a series of transformations on the structure on the report while preserving soundness and completeness of the logical content. The result is an automatically generated hypertext report that is both logically correct and, to a relatively high degree of confidence, free of misleading implicatures.

1 Introduction

Producing technical documentation is a time-consuming and expensive task. For instance, Reiter et al. (1995), report cases of engineers expending five hours on documentation for each hour spent on design and of airplane documentation sets which weigh more than the actual airplane being documented. Part of the reason for this problem is the gap between Computer Aided Design (CAD) tools and similar tools for assisting the documentation of those designs. Since research efforts focus primarily in the former, this situation is likely to get worse as the CAD tools get more powerful while documentation tools lag far behind.

In this paper we address the matter of automatic generation of technical documentation (Reiter et al., 1992; Reiter et al., 1995; Rösner and Stede, 1992; Svenberg, 1994; Punshon et al., 1997) by studying the problem of automatically generating documents describing the single step transition behavior of Statecharts.

From a natural language generation (NLG) perspective, this problem is distinguished in that the formal correctness of the document being generated is crucial while felicitousness of the style is relatively unimportant. This leads us to a solution based on formally verifiable theorem-proving techniques which allows us to approach strategic NLG issues within a highly abstract and conceptually clear framework.

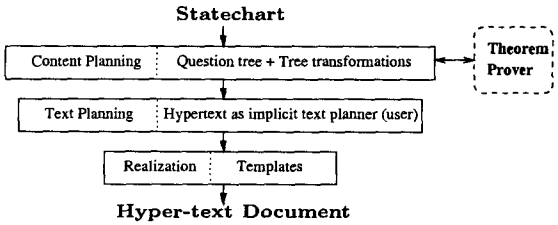

Figure 1: Conceptual view of the system.

The system takes a statechart in the form of a labeled directed graph and translates it into a set of propositional formulae defining its transition behavior. A hyper-text natural language document is generated on-demand from this set of formulae in response to the reader's interaction with the application.

Figure 1 depicts a comparative (Moore and Paris, 1993; Paris et al., 1991; Hovy, 1988) conceptual view of the system while Fig. 2 shows the system architecture. A prototype has been fully implemented with the exception of the statechart axiomatization module. [1]

2 A Logical Semantics for Statecharts

The graphical language of statecharts as proposed by David Harel (Harel et al., 1987; Harel and Naamad, 1996), has been widely recognized as a important tool for analyzing complex reactive systems. It has been implemented in commercial applications like STATEMATE (Harel and Politi, 1998)

[1] A full description of this algorithmic translation of a statechart from its graphical formalism to the propositional logic input format used in this work is described in Garibay (2000).

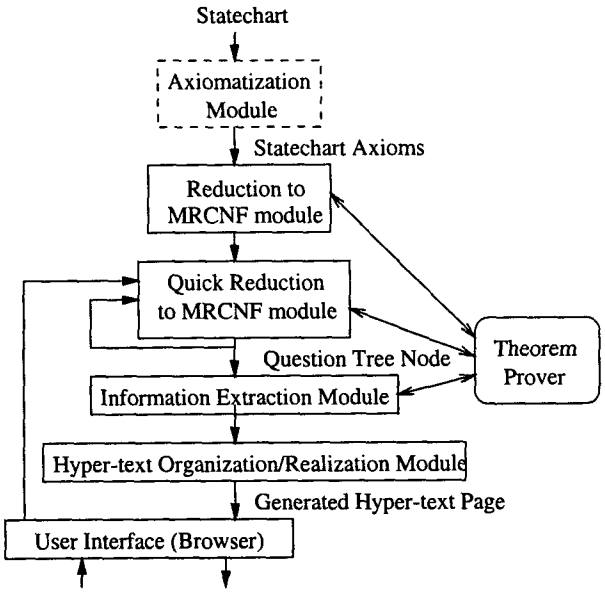

Figure 2: System architecture of the theorem prover based generator. The dotted box is not implemented.

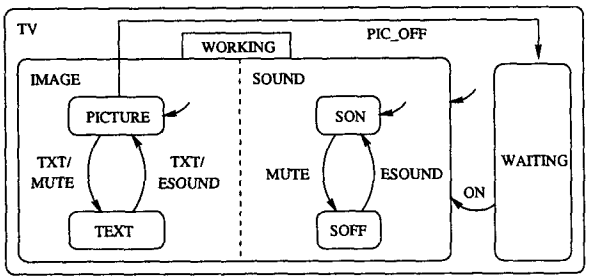

Figure 3: Example Statechart.

and RHAPSODY from ilogix (I-Logix Inc., 2000) and has been adopted as a part of the Unified Modeling Language (UML Revision Task Force, 1999; Booch, 1999), an endeavor to standardize a language of blueprints for software.

Statecharts (Fig. 3) are an extension of conventional finite state machines in which the states may have a hierarchical structure. A *configuration* is defined as a maximal set of non-conflicting states which are active at a given time. A *transition* connects states and is labeled with the set of events that trigger it, and a second set of events that are generated when the transition is taken. A *step* of the statechart relates the current configuration and the events that are active to the next configuration and the events that are generated. A configuration and the set of events that are active is referred to as a *status*.

We capture a step of a statechart as a pair of propositional models, one for the current status and

```
((TV ⟺ WORKING ∨ WAITING)
 (TV.next ⟺ WORKING.next ∨ WAITING.next)
 (WORKING → ¬ WAITING)
 (WORKING.next → ¬ WAITING.next)
 (WAITING → ¬ WORKING)
 (WAITING.next → ¬ WORKING.next)
 (WORKING ⟺ IMAGE ∧ SOUND)
 (WORKING.next ⟺ IMAGE.next ∧ SOUND.next)
...  )
((TV) ∧
 ((WORKING ∧ PICTURE ∧ PIC-OFF ∧ WAITING.next) ∨
  (WORKING ∧¬ (PICTURE ∧ PIC-OFF) ∧
      ((IMAGE ∧ PICTURE ∧ PIC-OFF ∧ WAITING.next) ∨
       (IMAGE ∧¬ (PICTURE ∧ PIC-OFF) ∧
          ((PICTURE ∧ PIC-OFF ∧ WAITING.next) ∨
           (PICTURE ∧ TXT ∧ MUTE.next ∧ TEXT.next) ∨
           (PICTURE ∧¬ OFF ∧¬ TXT ∧ PICTURE.next)
          )
...  )))
```

Figure 4: Section of the propositional logic translation of the example statechart (Fig. 3).

one for the next status. In practice, we incorporate this into a single model with two versions of each propositional variable: P for the truth value in the current status and Pn for the truth value in the next status [2]. A full description of the algorithm for translating statecharts to sets of formulae can be found in Garibay (2000). For a example of this translation see Fig. 4.

3 The Minimum Clausal Theory of the Statecharts

At this point, we have a formula that entails the theory of the single step transition behavior of a Statechart. We can fulfill our requirement of generating a sound and complete report just by translating this formula into English. However, this approach presents a number of problems. For instance, the AND and OR connectives do not in general have the same meaning in English as they do in logic (Gazdar, 1979), furthermore, unlike in the logical formula the scope of the connectives in English is not, in general, well defined (Holt and Klein, 1999). To minimize the ambiguity, we need to take the formula to a form with minimal nesting of operators.

Potentially a more significant problem is the fact that much of the theory (the formula plus all its logical consequences) is obtainable only via complicated inferences. Since the reader understands the translation of the formula at an intuitive level, making only limited inferences, a direct translation will fail to communicate the entire theory. Hence, we would like to take the formula to a form that is closed, in some sense, under logical consequences.

We address both issues by using what we refer to as *minimal (fully) resolved conjunctive normal form (MRCNF)*. A formula is in a MRCNF if and only if

[2] These single step models will form the basis for a temporal model capturing the full behavior of the statecharts as described by Harel and Naamad (1996).

it is in conjunctive normal form (CNF) and is closed under resolution, absorption and tautology (Fitting, 1990; Rogers and Vijay-Shanker, 1994). The closure under resolution is effectively a finite approximation of closure under consequence, that is, every clause that is a logical consequence of the theory entailed by the formula is a direct consequence of some clause in the MRCNF. The other two operations guarantee minimality in size by removing clauses that are trivially true (tautology), and those that are proper super-sets of another (absorption). Hence, the translation will communicate not only the initial facts but also those inferred by resolution. Moreover, a formula in this form is just a conjunction of disjunctions—eliminating the scoping problem. If we interpret the disjunctions as implications, the translation into English will be just a sequence of implicative sentences that are to be interpreted conjunctively—a typical structure for such information in English.

4 Organizing the Hyper-text Report: The Question Tree

A formula in MRCNF is organized in a way that resembles a sequence of implicative sentences. The problem now is the size of this sequence. Large to begin with, its size is increased by the transformation to CNF and closure under resolution. Hence, the translation of MRCNF directly into a sequence of statements would present an uninterpretable sequence of facts. If they are going to be understood by the reader there is a need for some kind of structure. The correct organization depends heavily on the reader's goals and expectations. However, beyond the assumption that the reader's generic goal is to obtain information about the transition behavior of the statechart under consideration, we do not make any assumptions about what the particular reader's goals may be. Instead we present the report as a hyper-text document and allow the reader to interactively refine their goal by following hyper-links. Effectively, the reader's queries focus the theory of the statechart in a particular aspect of its behavior[3].

In this way, as in Reiter et al. (1992) and Levine et al. (1991), we use hyper-text as an implicit text planner, in the sense that we account for every possible model of the user/system interaction and let the actual reader decide which goal to pursue.

We will call the reader's selections *choices*. Each choice the reader makes narrows the information we have to convey, limiting it to all and only the part that is logically consistent with that choice. We will say that the reader *refines* the theory by making the choice. At each point, the choices available to the reader are all the propositional variables that the theory is contingent upon. The reader effectively fixes the valuation of one of these variables to true or false. The system then adds the reader's choice to the theory and recalculates the MRCNF. If the newly obtained theory remains contingent upon some variables, the reader then will have available a new set of choices. If not, the reader will have reached a set of *non-contingent facts* (henceforth *facts*) which are consequences of all the previous choices.

While this process makes the information more accessible by giving it a logical structure, it does nothing to reduce the size of the report. We resolve this by generating the document on demand. While the refinement process (the core computation for on-demand generation) can potentially be very expensive in terms of time, the fact that we are adding singleton clauses to an already minimum set of clausal consequences allows us to use a simplified form of the theorem prover with asymptotic time complexity linear in the number of clauses.

We can visualize the process of the reader making choices as navigating a *question tree*, in which each branch is labeled with a choice and each node contains the theory of the Statechart as refined by the path of choices from the root to that node. In this tree, a reader's choice is equivalent to the question: "What are the circumstances/situations if X is true/false?." The root is the full theory of the transition behavior of the Statechart. The children of a node are obtained by fixing the valuation of each of its contingent propositional variables in turn and recomputing the MRCNF. The leaves are non-contingent theories (those containing only facts) [4].

Conceptually, the labels of each path from the root to a leaf together with each one of the facts in that leaf corresponds to all and only the valuations which are models of the original theory. Therefore, the question tree is sound and complete in the logical sense.

5 Generating the Hyper-text Page under Pragmatic Considerations: Information Extraction Module

This tree turns out to provide a useful framework to address pragmatic issues—those that arise principally from the structure of the report itself (Gazdar, 1979). By addressing these issues in the context of the question tree, rather than in its realization as a report, we abstract away from a great deal of subtle semantic detail that would otherwise obscure the analysis. Our approach consists of applying a series of transformations that resolve these issues while

[3] In a process that will be precisely described shortly.

[4] In general this structure is a directed acyclic graph which Reiter *et al.* call the *question space* (Reiter et al., 1995), but since we work with a tree that spans it, we prefer *question tree*.

preserving logical soundness and completeness of the document.

5.1 Promoting facts

In the question tree, the facts are either reported at the end of a chain of choices or are encoded in the choices themselves. A sequence of these choices is analogous to a chain of nested implications in which the antecedents are the choices made by the user and the consequence is the theory as refined by the choices. This refinement continues until we obtain a non-contingent theory—one in which all variables have valuations. Thus, the chain of implications eventually leads to a set of facts as its final consequence. The pragmatic problem in this case relates to the amount of information to be provided (Grice's *Maxim of Quantity* (Grice, 1975)). This maxim states that speakers will make their contribution as informative as is required, but not more informative than that (Gazdar, 1979). Under this assumption, reporting a fact as a consequence of a sequence of choices explicitly denies that this fact is a consequence of any prefix of that sequence, in contrast to the logical semantics of implication. Such implicatures, while not consequences of the logical content, are valid inferences that people make on the basis of well established expectations about the communicative act.

To avoid this false implicature, we present the facts to the reader as soon as they become available, that is, as soon as they become non-contingent in the theory. The transformation, in this case, moves the facts from the leaves to the interior nodes. This transformation does not change the set of models represented in the tree simply because the movement of facts does not eliminate any path of the tree. Hence, the transformation preserves soundness and completeness of the tree.

In practice, the facts are just the singleton clauses of a theory, therefore we can realize this transformation by simply reporting singleton clauses as soon as they appear in the theory.

5.2 Reporting facts only once

On the other hand, facts in a theory are also facts in every consistent refinement of that theory. Hence, reporting all the facts at each node of the question tree leads us to report many of them repeatedly. In effect, every fact reported in a node will be reported in each of its children as well. This repetition of facts violates the "upper-bound" of Quantity—it reports more than is relevant. In this case Quantity requires us to report only information that is "new".

In general, what is new will depend not only on what is reported but on inferences the reader is likely to have made (McDonald, 1992). We have, however, already committed to being explicit; our assumption is that the reader makes essentially no inferences, that they know all and only what we have explicitly reported. Therefore, we can satisfy the upper bound of Quantity by reporting each fact exactly once on each branch—when it first becomes non-contingent. To do this, we simply keep a list of all facts that have been reported in the current branch; this is the extent of our model of the user.

This transformation does not change the set of models represented in the tree, since it only eliminates repeated literals.

5.3 Promoting single level implications

One of the difficulties in using Quantity is to determine what information is "required". At each node of the question tree we have a current theory to report. The issue, in essence, is what to report at that node and what to report at its descendents. On one hand, it seems clear that we are, at least, required to report the non-contingent facts at each node. On the other hand, we don't want to report the whole theory at the root.

Our intuition is that the degree to which facts are relevant is inversely proportional to the difficulty of interpreting them. Under these circumstances, un-nested implications (i.e., binary disjunctions) are simple enough that the reader is likely to expect them to be reported. From the perspective of the question tree, this suggest, that in addition to the facts at a node, we should also report, as implications, the facts at its non-contingent children (those that are leaves). We refer to the choices leading to non-contingent theories as *conclusive choices.* These are reported as single-level implications ("If X then ⟨some sequence of facts⟩"). This has the effect of promoting the leaves of the tree to their parent pages.

Note that a choice that is conclusive at some page will also be conclusive at each page in the subtree rooted at that page (or, rather, at each page reached by a sequence of choices consistent with that choice). In keeping with the principle of reporting a fact exactly once along each path, we must avoid reporting the implication at the descendent pages. To this end, after reporting each of the conclusive choices on a page, we report the remainder of the tree below that page under an "Otherwise" choice in which the theory has been refined with the complements of the conclusive choices. This has the effect of dramatically restructuring the tree: each of the non-contingent leaves is promoted to the highest page at which the choice that selects it becomes conclusive.

Once again this transformation reorganizes the branches of the question tree without changing the set of models it represents.

To find the conclusive choices we run the theorem prover on the current theory extended, in turn, with each literal upon which it is contingent. If the resulting theory is non-contingent, then that literal is a

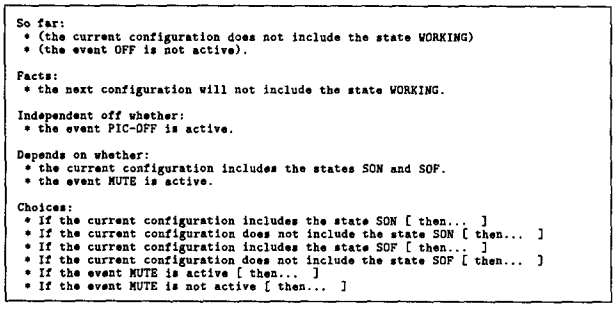

Figure 5: Example of generated hyper-text page.

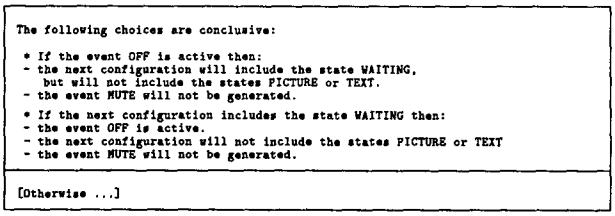

Figure 6: Conclusive choices section (up), non-conclusive otherwise section (bottom).

conclusive choice. To find the remainder of the tree to be reported under the "Otherwise" case we extend the current theory with the negation of each of the conclusive choices. If the resulting theory is inconsistent we will say that the conclusive choices are *exhaustive*, if the result is a contingent theory we will say that the conclusive choices are *non-exhaustive with non-conclusive otherwise*, and if the result is a non-contingent theory we will say that the conclusive choices, in this case, are *non-exhaustive with conclusive otherwise.*

5.4 Aggregating pairs of single conditionals

It frequently happens that, at some page, two conclusive choices lead to the same model. In this case, we would report that each implies (among other things) the other. However, these two implications can be aggregated to form a biconditional. Furthermore, Quantity requires us to select the strongest connective that applies in any such case because if a weaker connective is selected it suggests that no stronger one applies (a *scalar implicature*). Consequently, we are actually compelled to aggregate these two facts into a single biconditional.

In practice, we use the theorem prover to either prove or disprove, for every implication, whether its converse is a theorem of the current theory. If proved then the biconditional is reported.

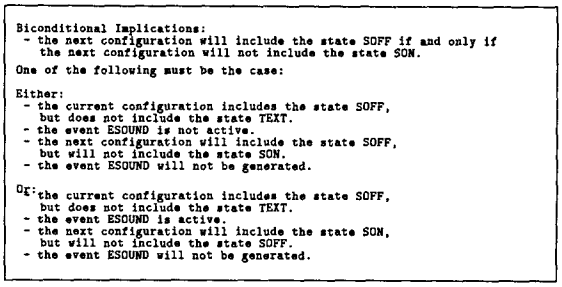

Figure 7: Biconditional implications and models sections.

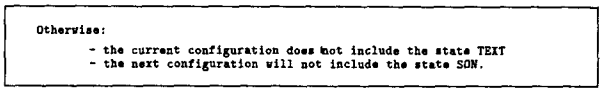

Figure 8: Conclusive otherwise section.

6 Hyper-text Organization and Realization Module

The organization of the hyper-text page generated from each node of the question tree visited by the user is shown in Fig. 5. At the top of the page we report (parenthetically) the set of choices that have led to this page. Next we report all of the new facts obtained from the current theory as described in sections 5.1 and 5.2. Then, the propositions that the theory is no longer dependent on (those which no longer occur in the theory) followed by the list of propositions on which it does depend. Finally we present the choices or, if there are any, the conclusive choices. In the first (Fig. 5), each choice is presented as an implicative sentence with a hyper-text link leading to another page (another node of the question tree). In the second (Fig. 6 top), we present the set of conclusive choices followed by one of the three possible cases (described in Section 5.3) for the "Otherwise" case. If the conclusive choices are exhaustive (the otherwise case is inconsistent), we report the biconditional implications (Section 5.4) followed by the final models (Fig 7). If they are exhaustive with a conclusive otherwise, we report the otherwise as another conclusive choice (Fig 8). Finally, if they are exhaustive with a non-conclusive otherwise, we report only an otherwise hyper-link (Fig 6 bottom).

The realization module is, in essence, a pattern matching and template filling process. It's basic component simply translates facts into fixed English language sentences.[5] Facts are represented by literals. These are classified into the following categories: *current state, current event, next state,* and *next event* and the literals in each category are syn-

[5]With added html mark-up.

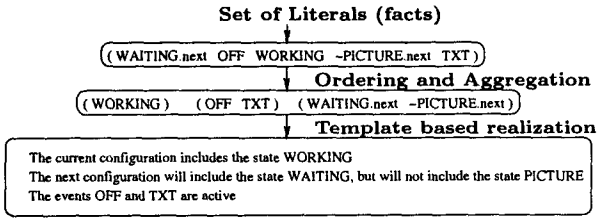

Figure 9: Example of realization.

tactically aggregated (Dalianis, 1999). The process is illustrated in Figure 9.

References

Grady Booch. 1999. UML in action. *Communications of the ACM*, 42(10).

Hercules Dalianis. 1999. Aggregation in natural language generation. *Computational Intelligence*, 15(4):384–414.

Melvin Fitting. 1990. *First-order Logic and Automated Theorem Proving.* Springer-Verlag, New York.

Iván Ibargüen Garibay. 2000. Automatic generation of natural language documentation from statecharts. Master's thesis, University of Central Florida.

Gerald Gazdar. 1979. *Pragmatics: Implicature, Presupposition, and Logical Form.* Academic Press.

H. Paul Grice. 1975. Logic and conversation. In Peter Cole and Jerry L. Morgan, editors, *Syntax and Semantics: Speech Acts*, volume 3, pages 41–58. Academic Press.

David Harel and Amnon Naamad. 1996. The STATEMATE semantics of statecharts. *ACM Transactions on Software Engineering and Methodology*, 5(4):293–333, Oct.

David Harel and Machal Politi. 1998. *Modeling Reactive Systems with Statecharts: The STATEMATE Approach.* McGraw-Hill. QA 76.9 .S88 H3677 1998.

D. Harel, A. Pnueli, J. P. Schmidt, and R. Sherman. 1987. On the formal semantics of statecharts. In *Symposium on Logic in Computer Science*, pages 54–64. Computer Society of the IEEE, Computer Society Press, June.

Alexander Holt and Ewan Klein. 1999. A semantically-derived subset of english for hardware verification. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics.*

Eduard H. Hovy. 1988. *Generating Natural Language under Pragmatic Constraints.* Lawrence Erlbaum Associates.

I-Logix Inc. 2000. http://www.ilogix.com.

John Levine, Alison Cawsey, Chris Mellish, Lawrence Poynter, Ehud Reiter, Paul Tyson, and John Walker. 1991. IDAS: Combining hypertext and natural language generation. In *Third European Workshop on Natural Language Generation*, pages 55–62, Innsbruck, Austria.

David D. McDonald. 1992. Type-driven suppression of redundancy in the generation of inference-rich reports. In R. Dale, E. Hovy, D. Rösner, and O. Stock, editors, *Aspects of Automated Natural Language Generation*, volume 587 of *Lecture Notes in Artificial Intelligence*, pages 73–88. Springer-Velag.

Johanna D. Moore and Cécile L. Paris. 1993. Planning text for advisory dialogues: Capturing intentional and rhetorical information. *Computational Linguistics*, 19(4):651–694.

Cécile L. Paris, William R. Swartout, and William C. Mann, editors. 1991. *Natural Language Generation in Artificial Intelligence and Computational Linguistics.* Kluwer Academic Publishers.

J. M. Punshon, J. P. Tremblay, P. G. Sorenson, and P. S. Findeisen. 1997. From formal specifications to natural language: A case of study. In *12th IEEE International Conference Automated Software Engineering*, pages 309–310, Incline Village, Nevada; USA, November. IEEE Computer Society.

Ehud Reiter, Chris Mellish, and John Levine. 1992. Automatic generation of on-line documentation in the IDAS project. In *Third Conference on Applied Natural Language Processing (ANLP-1992)*, pages 64–71, Trento, Italy.

Ehud Reiter, Chris Mellish, and John Levine. 1995. Automatic generation of technical documentation. *Applied Artificial Intelligence*, 9(3):259–287.

James Rogers and K. Vijay-Shanker. 1994. Obtaining trees from their descriptions: An application to tree-adjoining grammars. *Computational Intelligence*, 10:401–421.

D. Rösner and M. Stede. 1992. Customizing rst for the automatic production of technical manuals. In R. Dale et al., editors, *Aspects of Automated Natural Language Generation*, number 587 in Lecture Notes in Artificial Intelligence, pages 199–214, Berlin. Springer Verlag.

S. Svenberg. 1994. Representing conceptual and linguistic knowledge for multi-lingual generation in a technical domain. In *Proceedings of the 7th International Workshop on Natural Language Generation*, pages 245–248, Kennebunkport.

UML Revision Task Force, 1999. *OMG Unified Modeling Language Specification , v. 1.3. Document ad/99-06-09.* Object Management Group, June.