University of Brighton

itri-02-10

High-level authoring of illustrated documents

Kees van Deemter and Richard Power

Sept., 2002

Information Technology Research Institute Technical Report Series

ITRI, Univ. of Brighton, Lewes Road, Brighton BN2 4GJ, UK
TEL: +44 1273 642900 EMAIL: firstname.lastname@itri.brighton.ac.uk
FAX: +44 1273 642908 NET: http://www.itri.brighton.ac.uk

Natural Language Engineering **1** (1): 1–??. Printed in the United Kingdom

High-level Authoring of Illustrated Documents

Kees van Deemter and Richard Power
Information Technology Research Institute
University of Brighton, Lewes Road, Watts Building
Brighton BN2 4GJ, United Kingdom
{Kees.van.Deemter,Richard.Power}@itri.brighton.ac.uk

(*Received 24 July 2002*)

Abstract

This paper starts by introducing a class of future document authoring systems that will allow authors to specify the content and form of a *text + pictures* document at a high level of abstraction, while leaving responsibility for linguistic and graphical details to the system. Next, we describe two working prototypes that implement parts of this functionality, based on semantic modeling of the pictures and the text of the document; one of these two, the ILLUSTRATE prototype, is a *multimedia* extension of previous *text* authoring systems in the What You See Is What You Meant (WYSIWYM) tradition. The paper concludes with an exploration of the ways in which Multimedia WYSIWYM can be further enhanced, allowing it to approximate the 'ideal' systems that were sketched earlier in the paper. Applications of Multimedia WYSIWYM to general-purpose picture retrieval (in the context of the Semantic Web, for example) are also discussed.

1 Interactive Document Authoring

A *Document Authoring System* is a tool that helps an author to write documents. If the system supports the authoring of documents that combine presentations in different media (text and images, for example) we will, for want of a better name, speak of a *multimedia* document authoring system. Multimedia document authoring systems can be divided into low-level and high-level authoring tools.

Low-level document authoring tools allow authors to manipulate the physical constituents of a document: strings of characters, pictures, and so on. Such tools (*Word*, *PowerPoint*, *Dreamweaver*) are powerful but limited. They cannot guarantee the integrity (e.g., logical consistency) of the semantic content of the document, which is crucial in fault-critical applications. Also, they are of little help if the document has to be produced efficiently in different languages, or for audiences with different

presentational needs, or on platforms that support different combinations of media. A sophisticated variety of low-level tools, often based on XML/SGML, separates the conceptual and the physical layout of the document, allowing one conceptual description to be mapped onto different physical realisations. Unless and until, however, conceptual descriptions can express complex structural and logical relations, such tools are unable to capture formally the *meaning* of the document in detail, and consequently they share the above-mentioned limitations.

High-level document authoring tools allow authors to specify a document at a high level of abstraction, whereupon the system *generates* the document from this specification (Maybury and Wahlster 1998). Such 'Intelligent Multimedia Presentation Systems' (IMMPS, Bordegoni et al. 1997) have great potential for facilitating the authoring process, and for making it less prone to errors. In their present form, however, they have other drawbacks. Many of them are very limited as to what information the user can enter into the system; the menu-based interface of the COMET system is an example (Feiner and McKeown 1998). Other systems require input of a highly formal nature, making them difficult to use by domain specialists, who may not know anything about knowledge representation. A typical example is the WIP system, which takes fairly complex logical expressions as its input (André and Rist 1995). Moreover, they allow authors little control over the *form* of the document, and the degree to which they are able to combine text and other media properly into a coherent document is limited.

Ideally, a high-level multimedia document authoring tool would offer authors the following advantages:

1. *Easy determination of content.* 'Content' is the factual information in the document, which is encoded formally to allow inferencing (e.g., to enforce integrity constraints). Author should be allowed to enter content into the system as easily as possible. To a large extent, however, responsibility over the *expression* of this content should be taken away from the author and delegated to the system.

2. *Easy determination of style and layout.* In the absence of specific instructions, style and layout should be determined by the system using intelligent defaults. The author, however, should be allowed to override the defaults and make specific requests concerning the form of the document.
3. *Easy allocation of media.* Media allocation, too, should be handled using judiciously chosen defaults which may be overruled by the author. For example, the author might require that all and only information of a certain kind will be illustrated, or that one particular topic be illustrated by means of a picture.

In recent years, a technique called 'What You See Is What You Meant' (WYSIWYM) editing has been developed, which has been applied extensively to the problem of text authoring (Scott et al. 1998, Power and Scott 1998, Scott 1999). In this paper, we will argue that a variant of this authoring method can provide the advantages 1-3 mentioned above. As a preamble, section 2 describes an implemented system for the authoring of *textual* documents that can be argued to fulfill requirements (1) and (2). Section 3 describes two small, recently implemented prototypes in which significant aspects of requirement (3) have also been implemented. Most space will be devoted to the interactive ILLUSTRATE system, with a brief digression in section 3.3 concerning the RICHES document generation prototype (which is not interactive and does not use WYSIWYM). Section 4 explains what needs to be done to fill the gap between the implemented systems and the 'ideal' one described here. The paper concludes with an Appendix containing a worked example of the working of ILLUSTRATE.

2 A WYSIWYM-based System for the Authoring of Textual Documents

In a series of recent conference papers, a novel knowledge-editing method called 'WYSIWYM editing' has been introduced and motivated (Power and Scott 1998, Scott et al. 1998, Scott 1999). WYSIWYM editing allows a domain expert to edit a KB by interacting with a *feedback text*, generated by the system, which presents both the knowledge already defined and the options for extending and modifying it. Knowledge is added or modified by menu-based choices which directly affect the

KB; the result is displayed to the author by means of an automatically generated feedback text: thus 'What You See Is What You Meant'. WYSIWYM editing is a sophisticated instantiation of a more general idea called 'symbolic authoring', which involves a Natural Language Generation system producing a text on the basis of a formal specification (in the form of a symbolic KB). The advantages of this approach are comparable to those of creating, for example, a silicon chip, from formal specifications rather than on the drawing board: the process becomes less error-prone, and the product is more easily updated. The advantages are most pronounced if the text (or the chip) has to be updated regularly and if texts in different languages, or for different audiences, have to be produced. WYSIWYM makes symbolic authoring *easier* by offering the author written feedback, in his or her own language, concerning the state of the KB and the ways in which it may be modified. By using language generation rather than language interpretation, WYSIWYM instantiates a recent trend in dialogue systems towards moving some of the *initiative* from the user to the system, allowing such systems to avoid the difficulties of processing 'open' (i.e., unconstrained) input.

Of particular importance, in this paper, are applications of WYSIWYM to the generation of documents containing text and *pictures*; the present section focuses on (multilingual) *text* generation: the KB created with the help of WYSIWYM is used as input to a natural language generation (NLG) program, which produces as output a document of some sort for the benefit of an end user. Present applications of WYSIWYM to text generation use a KL-ONE-type knowledge representation language as input to two NLG systems. The use of these representations implies the standard distinction between a Terminology Box (T-Box), which legislates what are wellformed expressions of the KR language, and the Assertions Box (A-Box), which contains a set of expressions that the T-Box determines to be wellformed (e.g., McGregor and Oates (1987). One of the two NLG systems generates feedback texts (for the author) and the other generates output texts (for an end user). One application currently under development has the creation of Patient Information

Leaflets as its domain. A preliminary version of this system has been developed for the ICONOCLAST project.[1] ICONOCLAST allows authors to enter information about possible side effects (e.g., *'If you are either pregnant or allergic to penicillin, then tell your doctor'*) and how to handle medical devices such as inhalers, inoculators, etc. By interacting with the feedback texts generated by the system, the author can define a procedure for performing a task, e.g. preparing an inhaler for use. A new KB leads to the creation of a procedure instance, e.g. p. The T-Box specifies which procedures are complex and which atomic, and lists a number of options in both cases. In the atomic case, the options include `Clean, Store, Remove`, etc., and these are made visible by means of a menu from which the author can select, say, `Remove`. The program responds by adding a new instance, of type `Remove`, to the KB:

$Remove(p)$

('There is a procedure p whose type is `Remove`.') From the updated KB, the generator produces a feedback text

Remove **this device** from **this device**,

making use of the information, in the T-Box of the system, that a `Remove` procedure requires an `Actee` role and a `Source` role to be defined. (Henceforth, the role of the `Actor` of an action will be suppressed.) Not yet defined attributes are shown through mouse-sensitive anchors.[2] By clicking on an anchor, the author obtains a pop-up menu listing the permissible values of the attribute; by selecting one of these options, the author updates the KB. Clicking on **this device or device part** yields a pop-up menu that lists all the types of devices and their parts that the system knows about, including a Cover (which, according to the T-Box must have a Device as `Owner`). By continuing to make choices at anchors, the author might expand the KB in the following sequence (for elaboration, see Appendix):

[1] ICONOCLAST was supported by the EPSRC under grant L77102. A follow-up system with larger linguistic coverage has been built for the PILLS project (ECD-3310-26904).

[2] The system uses colour coding to indicate the difference between optional (green) and obligatory (red) attributes. When no more red anchors remain, we shall say that the KB is potentially complete.

- Remove **this device**'s cover from **this device**
- Remove **this person**'s inhaler's cover from **this device**
- Remove your inhaler's cover from **this device**
- Remove your inhaler's cover from your inhaler

After this editing sequence the KB is potentially complete, so a (less stilted) *output text* can be generated and incorporated into the leaflet, e.g.,

> Please remove the cover from your inhaler.

Longer output texts can be obtained by expanding the feedback text further.

A number of properties of the ICONOCLAST system[3] are worth stressing. First, the system supports a high-level dialogue, allowing the author to disregard low-level details, such as the exact words used in the output text. This makes it possible to interact with the system using, say, French (provided a generator for French *feedback* texts is available), for the production of leaflets in Japanese (provided a generator for Japanese *output* texts is available). The semantic model in the T-Box guarantees that many types of inconsistencies (e.g., a medicine that has to be taken both once and twice a day) are prevented.

Second, the user can control not only the language but the *style* of the output texts, by specifying preferences about such features as sentence length, use of semicolons, use of technical terms (e.g., *larynx* in preference to *throat*), and the prevalence of graphical layout devices like bulleted lists. Both these properties of the ICONOCLAST system exploit the basic WYSIWYM concept of authoring as content specification: once the desired meaning has been encoded in a KB, the system can automatically generate versions that vary in presentational features like language, style, and also – as we shall see – in the possible inclusion of illustrations.

WYSIWYM editing has been used successfully in a number of research projects,

[3] ICONOCLAST is implemented in Eclipse Prolog, using ProFIT (Erbach, 1995) as a means of encoding feature structures. The user interface is implemented in the Common Lisp Interface Manager (CLIM 1994).

including ICONOCLAST, PILLS, and CLIME (Computerised Legal Information Management and Explanation, funded by the ESPRIT programme, EP25414). Standard User-Interface questionnaires and analytical studies indicate that the interface is useful for novice users, although there are also a few weaknesses to be ironed out (Masthoff 2000). In particular, recent formative studies using Cognitive Walkthrough and Nielsen-style Heuristic Evaluation suggest that WYSIWYM-based systems can have 'bootstrapping' problems (i.e., when the initial feedback text comes up, which is informationally almost empty), which may be countered by the addition of error messages or by showing authors a 'live' example of how the interface can be used (see e.g., `http://www.itri.bton.ac.uk/research.html#ICONOCLAST`). Problems are also in evidence when texts become extremely large, making it difficult to view all the necessary information on one screen. Perhaps most importantly, the system is not yet able to detect some of the subtler inaccuracies that can occur in the KB. ICONOCLAST, for example, would allow authors to specify that a cover can be removed by removing the cover (which may be accomplished by removing the cover, etc.). Future versions of the system should allow fewer and fewer such inaccuracies, by applying more and more integrity constraints to the content of the KB. In what follows, we assume that such improvements will be possible, and we explore the application of WYSIWYM-style document authoring to a more complex type of documents.

3 A WYSIWYM-based System for the Authoring of Multimedia Documents

Here we introduce an extension of WYSIWYM, called *multimedia* WYSIWYM, which allows the authoring of multimedia documents. The approach will be exemplified using the ILLUSTRATE system, which is a novel extension of ICONOCLAST. ILLUSTRATE produces documents that contain pictures as well as words. The application domain of the system is the same as before: patient information leaflets.

Pictures occur in about 60% of the information leaflets of the Association of British Pharmaceutical Industries Compendium (ABPI 1997); all leaflets in the corpus con-

tain text as well. For concreteness, we have focused on all the leaflets of one company. Because we were specifically interested in pictures of *actions*, (because these tend to be structuraly complex) we chose a company, Boehringer Ingelheim, that specializes in anti-asthmatic or anti-congestion inhalers, which tend to come with complex instructions for use. We constructed an interface that would help a person who has to author each of the 25 leaflets of this company, 21 of which contain pictures. The leaflets actually used by the company contain a total of only 31 pictures, most of which are used in a number of different leaflets. Consider a (slightly adapted) example. The text says *Remove the cover of your inhaler*, and this is illustrated by the picture below. How can a document authoring system produce

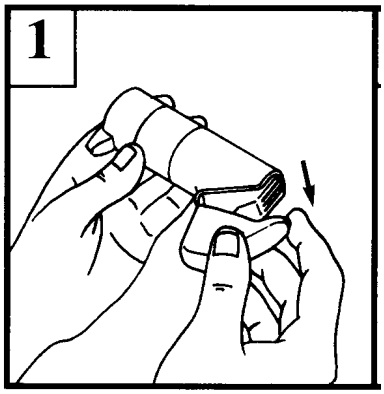

Fig. 1. One of the pictures in the library of the authoring system

a document in which appropriate pictures illustrate the text when this is desired? ILLUSTRATE does this by allowing an author to ask for pictorial illustration of the information in the document by interacting with the feedback texts. The author can indicate, for a given mouse-sensitive stretch s of the feedback text, whether she would like to see the part of the document that corresponds to s illustrated. If so, the system searches its library to find a picture that matches the meaning of s. In Fig.2, the author has requested illustration of the instruction corresponding with the text 'Remove your inhaler's cover from your inhaler'. (The other four options are irrelevant for present purposes.) In domains where all the pictures are variations on a common theme, suitable pictures can often be *generated* (e.g., Wahlster et al.

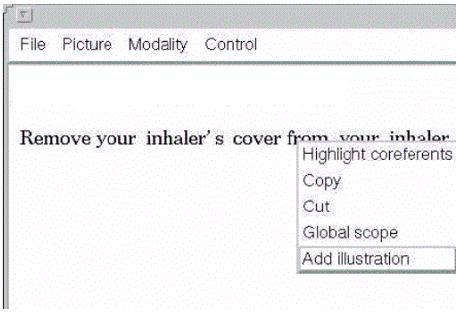

Fig. 2. Screendump: Author makes a requests for illustration

1993, André and Rist 1995). In the case of Patient Information Leaflets, however, this was not a practical option because of the many different kinds of things depicted in the leaflets: medicine packages, body parts, medical applicances, various types of actions, etc., each of which have different 'vocabularies' and pictorial styles. Pictures, moreover, are heavily reused: each picture is used 3 times on average. In fact, the number of *different* pictures used by any of the companies represented in the corpus is always very limited (always below 100). For these reasons, ILLUSTRATE uses an alternative to picture generation, *retrieving* pictures from a library. (In other words, we assume that the person authoring the leaflets has a limited number of pictures to choose from, including the ones that occurred in the actual leaflets.) We will explain the workings of ILLUSTRATE by answering three questions: **(1)** What kinds of representations are used in the library to annotate (or index) the pictures with relevant aspects of their meaning? This will be discussed in section 3.1. **(2)** What retrieval algorithm is employed to retrieve an optimally appropriate illustration for a given part of the KB from the library? (See section 3.2.) And **(3)** How are pictures integrated with the rest of the document? (Section 3.3.) We shall assume that the information whose illustration is requested corresponds with the following formula in the KB, which represents the meaning of the feedback text in Fig.2.

$$Remove(p)\ \&\ Source(p) = y\ \&\ Inhaler(y)\ \&\ Actee(p) = z\ \&$$
$$Cover(z)\ \&\ Owner(z) = y\ \&\ Owner(y) = u\ \&Patient(u).$$

('There exists a 'Remove' action whose `Source` is an Inhaler and whose `Actee` is a Cover of the same inhaler, while the `Owner` of the inhaler is the patient.') We shall call this the *target* information. Recall that, when the author requests illustration of the target information, a textual version of this information has already been generated but, if necessary, the system can generate an alternative text to match the illustration (see section 4.2).

3.1 Meaning representation and interlinguality

To relate the text of the document to an appropriate illustration we need a basis for comparing the two. Let us simplify and assume that the suitability of a picture depends on the factual information conveyed by each of the two alone. Thus, the wording and even the language of the text are irrelevant, and so is the rendering of the picture. On this assumption, the problem can be reduced to that of linking the picture with the *target* information in the KB. This does not solve the problem until a formal representation of the meaning of the picture, in a form compatible with that of the target information in the KB, is also available. A representation language whose expressions are compatible accross two media (e.g., text and pictures) has been dubbed a multimedia *interlingua* (Barker-Plummer and Greeves 1995).

While studying the way in which pictures are used in ABPI (1997) we discovered that, in this domain at least, a multimedia interlingua is not difficult to find, since pictures express largely the same *kind* of information as text, only in a way that is sometimes more detailed, sometimes less. The text, for example, might instruct the reader to *'Apply some cream to the blister'*, while the picture shows the same information, plus the fact that this is done *by hand*. This suggests that pictures might be represented using the same language that is employed to create the KB, and this is the approach chosen for the ILLUSTRATE system. Suppose the original T-Box allowed one to represent the meaning of the example sentence as

$$Apply(p)\ \&\ Actee(p) = x\ \&\ Cream(x)\ \&\ Destination(p) = y\ \&\ Blister(y).$$

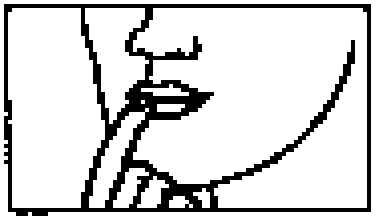

Fig. 3. *Apply some cream to the blister*

('There exists an Apply action whose `Actee` is a Cream and whose `Destination` is a Blister'.) Then an extended T-Box might attribute a 'Tool' feature to an action of this kind, allowing one to specify in more detail how the action is performed:

$Apply(p)$ & $Actee(p) = x$ & $Cream(x)$ & $Destination(p) = y$ & $Blister(y)$ & $Tool(p) = z$ & $Hand(z)$.

In fact, the picture shows even more detail (e.g., the index finger is used), but the question is whether this is relevant to the intended meaning of the picture. We assume that the designer of the T-Box (i.e., the person who defines the interlingua) knows which details matter, in this type of document.[4] If it matters which finger performs an 'Apply' action, then the T-Box has to become even more refined, causing the just-proposed representation to become insufficiently detailed. At an even more refined (and less plausible) level, the fact that the person depicted does not have a beard might or might not be included in the representation.

The fact that textual and pictorial information must be compatible does not mean they have to be indistinguishable. In important respects, for example, pictorial information is simpler than textual information: as observed in Levesque (1986), pictures tend to express 'vivid' information, that is, information expressible by a conjunction of positive literals. Pictures are bad at expressing logically structured information such as, for example, disjunctions, and quantification. Levesque's claim is confirmed by the pictures in our corpus: the only strictly logical information expressed through pictures is *conjunctive*, through juxtaposition (though only in an implicit way that makes conjunction indistinguishable from temporal succession)

[4] This assumption is not always unproblematic. See section 3.3 for discussion.

and sometimes *negative*; negation, however, is expressed only a few times (by a cross through a picture), and these few cases are safely ignored for now.

Returning to our original example, ILLUSTRATE represents the meaning of the picture in Fig.1 as follows:

$$Remove(p)\ \&\ Source(p) = y\ \&\ Haler(y)\ \&\ Actee(p) = z \& Cover(z)\ \&\ Owner(z) = y.$$

(*In*halers, *Auto*halers, and *Aero*halers have many properties in common, hence the underspecified property 'Haler' which generalises over these three.) If any of the variables p, y, z has an occurrence in the meaning representation of another picture then these occurrences corefer. This allows the system to know when two pictures depict the same person, for example (Van Deemter and Power 1999).

What we have explained so far is the construction of a T-Box. An important practical question, however, is how to link a picture to an A-Box representation that conforms with the T-Box. Fully automatic construction of the semantic representation of a picture is not yet feasible, given the difficulties involved in picture recognition. We will, therefore, assume that representations are created with human input.

The answer to this question may be unexpected: Indexation involves the creation of logical content, and can therefore be viewed as the creation of an A-Box, distinct from, but analogous to, the creation of the 'textual' A-Box. Hence, a WYSIWYM interface can be employed to create indexations. Fig.4 shows a screendump of the indexation process, where the current, incomplete indexation equals the formula

$$Remove(p)\ \&\ Source(p) = y\ \&\ DeviceorDevicepart(y)\ \&\ Actee(p) = z\ \&\ Cover(z)\ \&\ Owner(z) = x\ \&\ Device(x).$$

('There is a Remove action whose `Source` is an object y and whose `Actee` is the cover of an object x'.) Note that this formula is highly unspecific. The ILLUSTRATE T-box requires the identity of the `Source` of a removal action to be specified, which is why the interface treats the annotation as being incomplete (i.e., not even *potentially* complete), which is shown by the fact that the phrase *'a device or device part'* appears in red in the feedback text (see footnote 2). When the annotation is finished, the corresponding feedback text could be equal to that in Fig.2. The

top of the screendump shows the feedback text containing anchors for further additions. This feedback text is generated on the basis of the formula displayed above. Indexation might be speeded up further if content-based retrieval techniques (Ven-

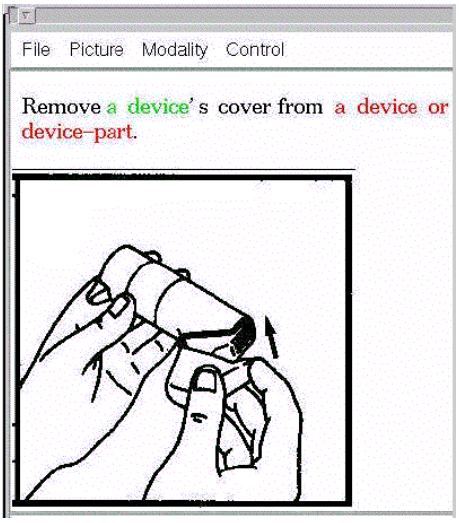

Fig. 4. Screendump: A stage during the indexation of a picture

ters and Cooper 2000) are employed to generate a first approximation of such representations, to be revised and completed by human intervention. Current prototypes, however, let authors use WYSIWYM to build up picture representations from scratch.

3.2 Retrieval of appropriate illustrations

Given that every picture is now associated with a semantic representation, how are pictures matched with the target information? Papers on existing document authoring tools, such as the Editors' Workbench of Bateman et al. (2001), appear to have little to say about this problem, sometimes focussing on the choice between text and schematic diagrams instead.[5] In the present section, we will explain how our semantic representations allow us to select pictures in highly controlled fashion.

[5] Bateman et al. (2001), for example, say '(...) the photo option is self-selecting – i.e., if information is only available as a picture then that is what will be slotted into the layout structure', leaving unexplained what information *is* expressed by pictures.

We have seen that a picture can depict things that are irrelevant to its meaning. For example, whether or not a person has a beard is irrelevant in a document on cold sore cream. This is captured by the limited expressivity allowed by the T-Box of the system. (For example, we have assumed that the property of having a beard is not expressible.) Conversely, however, a picture can illustrate an item of information without expressing every aspect of it. For example, Fig.1 leaves the type of 'Haler' unspecified. (All 'Halers' look alike.) Consequently, a retrieval rule must allow pictures to omit aspects of the target information. The *more* relevant information is omitted, however, the *less* appropriate the illustration becomes as an illustration of the target information. In Van Deemter (1998, 1999), a number of different retrieval rules are discussed, and the following is proposed as most appropriate:

Retrieval Rule ('Rule B'): *Use the logically* **strongest** *picture whose representation is logically* **implied by** *the information to be illustrated.*

Logical strength is determined on the basis of the two semantic representations alone, along familiar lines. If more than one picture fulfills **Rule B**, the one with the largest number of conjuncts is chosen, and if this leaves the system with any ties, a picture is chosen arbitrarily. (Future versions of the system may leave the choice between equally preferred pictures to the author.) Let us return to our example, with

Target information:
$Remove(p)$ & $Source(p) = y$ & $Inhaler(y)$ & $Actee(p) = z$
& $Cover(z)$ & $Owner(z) = y$ & $Owner(y) = u$ & $Patient(u)$.

Picture meaning (Fig.1):
$Remove(p)$ & $Source(p) = y$ & $Haler(y)$ & $Actee(p) = z$ & $Cover(z)$ & $Owner(z) = y$.

Rule B checks that the meaning of the picture is logically implied by the target information, concluding that Fig.1 is a *possible* illustration of the target information. (Another picture may be preferred if it also is a possible illustration, while being more informative than Fig.1.) In our example, the job of **Rule B** is easy, since the two representations happen to use the same variables (p, y, z); if this is not the case, a suitable mapping has to be found between the variables.

Note that the information selected may be logically related to non-selected information, in which case the related information should be counted as part of the target information. For example, suppose the KB contains information of the form $p \rightarrow q$, whereas only (this occurrence of) q is selected; in this case, a picture may illustrate both p and q. To determine which information counts as related to the information selected, we use the notion of logical 'accessibility', one of the key notions to emerge from Discourse Representation Theory (e.g., Kamp and Reyle 1993), to say which bits of information, in a logically structured expression, are dependent on which others. Thus, the information in an antecedent is *accessible* from the consequent, but not the other way round. Accessible information counts as part of the target information, as long as it is relevant to the information selected. To clarify the notion of accessibility, imagine that the KB contains an implication of the form $p \rightarrow (q \rightarrow r)$, while only r is selected for illustration. Then an illustration of r can also depict information from q, and even p, if it is relevant. Since antecedents are accessible from consequents, but not the other way round, then if p is selected, the most informative illustration will express no more than p. If q is selected, then the most informative illustration expresses $p\&q$.

For concreteness, suppose the selected information is r = *'apply elixir cream to your face'*; suppose, furthermore, p = *'you have a blister on your lip'*, and q = *'your face is red'*, then if r is selected, it will be appropriate to identify $p \rightarrow (q \rightarrow r)$ as the target information and to look for pictures of *red* faces with a *blister*, to which a *cream* is being applied. The information in p and q is relevant to that in r because it expresses properties of an object (i.e., the face) mentioned in r.

The ILLUSTRATE prototype allows only simple conditions (events, states, or conjunctions/sequences of events or states) to be selected for illustration. If this limitation is removed, some interesting complications can arise. Suppose, for example, the KB fragment selected takes the form of a disjunction, $p \vee q$. Then one can argue that there are several different targets: a (maximally informative) illustration of p would be welcome, but so would a (maximally informative) illustration of q. A third

possibility arises if a picture is suitable for illustrating the disjunction as a whole. Imagine, for example, that $p = Man(x)$ and $q = Woman(x)$. Then if the library contains a 'gender neutral' picture expressing $Person(x)$ (which is equivalent to $p \vee q$), then **Rule B** would favour this picture as the most informative picture illustrating the disjunction. Similar complications arise with other logical operators: for example, if a negated proposition, say $\neg p$, is selected, then illustrations of $\neg p$ are appropriate but, in some domains at least, so are illustrations of p itself. (For example, when a patient information leaflet warns *against* children getting hold of medicines, this is sometimes accompanied by a picture showing children opening the medicine box.) More extensive corpus research is needed to determine how logically complex information is best illustrated.

3.3 Some problems with Illustrate

A more detailed example of the working of the ILLUSTRATE interface can be found in the Appendix. Although ILLUSTRATE proved to be a considerable help in annotating the Boehringer Ingeheim pictures, we also found a number of limitations and problems. We briefly mention three related issues.

1. **Rendering** of pictures. Sometimes, 'the same' picture is rendered in different ways on different occasions. For example, their sizes and foregrounding properties may be different, and details may be more or less visible. As long as these differences do not affect the meaning of the picture, they fall outside the system's ability to control them, which makes it difficult to ensure consistency of pictorial style throughout the document, except where the properties involved can be changed by the system (see, e.g., section 3.4).
2. **Consistency** within sequences. Consistency is even more important for pictures that belong together in a tightly knit sequence (illustrating, for example, a sequence of instructions for performing a complex task). If the first picture in the sequence depicts a man, for instance, and the second a woman, then this will be taken to imply that they are different people, whereas a proper understanding may require them to be interpreted as the same person. This is

a tricky problem, because consistency extends beyond the properties that the T-Box makes expressible (hair colour, length, complexion, etc.) One element of the treatment of consistency within sequences is the sharing of variables accross representations, which allows annotators to express whether two depictions can be interpreted as referring to the same object (Van Deemter and Power 1999).

3. **Knowing what matters**. Let us follow up on the example of gender. If a picture shows a woman, should the annotator mark her gender? If gender never matters, in any of the leaflets of the company – as was the case for the Boehringer Ingelheim leaflets – then, obviously, the annotator should ignore it, in which case case, the T-Box will forbid adding it to an annotation. But suppose the company makes contraceptives as well, which makes gender a relevant property in some cases. Now suppose the annotator works on a picture of a woman holding an inhaler. In this case, the T-Box has to allow the specification of gender (since it is relevant for *some* pictures), yet gender is irrelevant in this specific case. If gender is accidentally specified to be female (the person can be seen to be a woman, after all) then, as a result, **Rule B** will fail to find this picture when looking for a '*person* holding an inhaler'. In other words, the annotator has to know which of the properties depicted as holding for the objects in a picture are relevant.

3.4 Integration of pictures into the document

The ILLUSTRATE prototype has an extremely simple treatment of issues relating to physical and linguistic integration of pictures and text. A slightly more principled approach was followed in the RICHES system (Cahill et al. 2000). The RICHES system was designed to explore the relations between different levels of document structure – from abstract rhetorical structure all the way to the layout – and pictures are taken into account at all these levels. Pictures make their first appearance in RICHES when **Rule B** causes Rhetorical Structure to be modified, by stipulating that a certain item of information will be illustrated by a certain picture (i.e., a `gif` file on the

world-wide web). Unlike ILLUSTRATE, RICHES does not allow authors to control picture location. Instead, the system postions the pictures 'autonomously': at the first suitable position following the *first* clause in the document for which it is a best-matching illustration. (Suitable positions are taken to be those that coincide with a paragraph break.) When a location for a picture has been found, RICHES' Rendering Module changes the size of the picture if necessary.

4 Beyond the ILLUSTRATE system

In this section, we will chart what could be done to bridge the gap between the 'ideal' document authoring system described in the Introduction and the small prototypes we have developed (in particular the ILLUSTRATE system).

4.1 Bridging the gap: enhancing ILLUSTRATE

Even though its linguistic coverage is modest, the ICONOCLAST system (section 2) goes some way towards fulfilling text-related requirements 1 and 2 mentioned in section 1. The ILLUSTRATE demonstrator also addresses requirement 3, which relates to a proper treatment of nontextual media. The design of a fully fledged document authoring system is a huge and multifaceted task, however, and there is still a considerable gap between the implemented system and the ideal one outlined in section 1. Possible improvements include the textual/pictorial quality of the documents, and the interactive properties of the system. In addition, the coverage of the system could be enhanced by importing aspects from other systems. These issues include, for example, the generation of graphics from underlying representations (Wahlster et al. 1993) and the problem of optimizing the layout of text & picture documents (e.g., Graph et al. 1996). The same is true for the generation of captions (Mittal et al. 1998). Alternatively, an extension of the present approach will allow us to *generate* captions from small, separately-created KBs (i.e., one for each picture). Prominent other possibilities include the following:

1. *Treating authors as readers and readers as authors.* Authors are also readers. WYSIWYM-based authoring tools allow one author to read (and adapt) what

another author has written, even if they do not share a language. But even an *ordinary* reader may want to influence the form of the document. *Low-level* tools for the authoring of electronic documents (cf. section 1) can support this by offering a reader control over font and page size, for example, and sometimes (on the web) a choice between two or more languages, and presence/absence of graphics. *High-level* tools like ILLUSTRATE offer the reader more fine-grained control, almost as if they were authors, since readers can use the same menus as authors. In a further extension, readers might even be given limited control over the *content* of the document. For example, they may request that a designated part of the document be summarized.

2. *Allocating Media.* ILLUSTRATE embodies one way in which media may be allocated. Other mechanisms can be employed to give the system more autonomy. A simple example of this approach was mentioned above (section 3.3) in connection with the RICHES system; for a combination with graphics generation, see Roth and Hefley (1993). Similarly, authors may be enabled to point at thumbnail pictures, whereupon the system tries to find a suitable place in the document to include them, based on the representation of their meaning and making use of **Rule B** of section 3. By thus allowing the author and the system to cooperate on media allocation, this difficult task will be made more tractable (ETAI 1997-8).
3. *Treating text as a picture.* Little in ILLUSTRATE hinges on the fact that the objects in the library are pictures. The same interface can be used for annotating canned text, sound, or video. Of great practical interest, finally, is the possibility of including documents authored previously (and possibly by a different author), leading to iterative applications of WYSIWYM. It might be possible to apply the same principles to the generation of *animated* presentations (e.g., Zhou and Feiner 2001), but this possibility must be left unexplored here.
4. *Adapting text.* So far, we have said nothing about the way in which the text of a document is affected by the addition of pictorial illustrations: after dis-

cussing text authoring in section 2, we moved on to the inclusion of pictures, pretending that the texts remained unaffected. Note, however, that this is not necessarily the case. Illustration, in our approach, is primarily an operation on a KB: it's the *target information* that **Rule B** connects with the representation of an appropriate picture, not the text that *expresses* the target information. Nothing prevents the system from using the illustrated KB to generate a revised text, which takes the illustrations into account.

The last-mentioned issue will be discussed in more detail in the next section.

4.2 Interactions between text and pictures.

One of the main advantages of our integrated approach to text and pictures is that it allows us to vary the text depending on the extent and the manner in which it is accompanied by illustrations. We will briefly discuss two ways in which text can be influenced by pictures: First, we will discuss references to illustrations (as in 'See Fig.3.'). Secondly, texts may be reduced because of information expressed in the picture.

References to illustrations. References to illustrations can help to improve the readability of the text, by highlighting a connection between two or more of its parts. A sentence like 'Compare Fig.7 of the previous section', for example, makes crucial use of such a reference. Our semantic approach to pictures allows us to generate such references, whether they are based on the location of the picture (as in 'See Fig.7') or on its pictorial content (as in 'See the picture *of the inhaler*'). Conversely, the fact that a picture depicts a certain domain object can also be exploited for identifying the object, as in 'the red vial depicted in Fig.7'. Descriptions of this kind can be generated using a variant of standard generation algorithms (e.g., the Incremental Algorithm of Dale and Reiter 1995) provided document-related properties like 'being described by Fig.7' are treated on a par with other properties (such as being red, being a vial, etc.) This is only possible in a system that 'knows', of every picture in the document, which domain object(s) it refers to. Note that this

information is not always deducible from the semantic annotation of the picture alone, since a given picture can refer to different objects. Algorithms for automatically generating 'document deictic' references of this kind are discussed in Paraboni and van Deemter (2002), and have been implemented in a small stand-alone generation program by Ivandré Paraboni.

Reducing text. Something that is expressed through a picture may no longer have to be expressed textually: information may be omitted from the text if this does not result in a loss of information. One type of situation where this happens arises when quantitative information is expressed through a *vague* textual description ('a blob of cream', 'a fingertip of ointment') that is made more precise by means of a picture that graphically displays the required amount (ABPI 1997). For another type of example, let us return to Fig.3, which involved applying some cream to a blister. Suppose the KB required that the cream was applied using the hand, and that this was expressed by the textual instruction *'Use your hand to apply some cream to the blister'*. If Fig.3 illustrates this information then the fact that the hand is used becomes textually redundant. If this information is removed, for purposes of text generation, then the result equals the earlier, simpler formula:

$$Apply(p) \;\&\; Actee(p) = x \;\&\; Cream(x) \;\&\; Destination(p) = y \;\&\; Blister(y),$$

This might allow the generator to produce the original, simpler text, which said *'Apply some cream to the blister'*. Given the structure of Description Logic, which allows the distinction between Role and Type clauses, we have to distinguish between *removal* of the *Role* of an Instance, on the one hand, and *generalization* of the *Type* of an Instance. An example of the former occurred above, where the `Tool` role was removed. An example of the latter would occur if, for example, a Type clause '$LittleFinger(x)$' was replaced by a more general clause, say '$Finger(x)$'. Overlap between the information conveyed by text and pictures is not necessarily a bad thing, and the optimal amount of overlap depends on genre and style. On the other hand, the reduction of information is not unconstrained, or else a representation may result that is either illformed or misleading. For example,

- Information of the form '$Role(x) = y$' cannot be removed from a representation if the resulting representation has any remaining information of the form '$Role(y) = z$'. (If this happened, '$Role(y) = z$' would no longer be connected to the rest of the representation.)
- Information of the form '$Type_1(x)$' cannot be replaced by Information of the form '$Type_2(x)$' if any of the remaining propositions of the form '$Role(x) = y$' becomes illformed as a result. (This happens if y is a legitimate role of $Type_1$ but not of $Type_2$.)
- Information should not be removed from within a negation or a conditional. Suppose a document says *'If you have an extremely serious headache then take five pills'*, illustrated by a picture of someone with an extremely serious headache. Simplification of the text to *'Take five pills'* would invite the incorrect interpretation that five pills need to be taken regardless of whether one has an extremely serious headache.

The idea that document authoring systems should take interactions between text and pictures into account is not new. Some existing systems, for example, allow the text to omit information about the location of a device if this location is made visible by a picture (André and Rist 1995). The constraints described explain why such rules are valid. Cases of the kind discussed by André and Rist (1995), for example, can be analysed as follows:

- **KB:** TurnUp(p) & Actee(p) = x & TempControl(x) & Loc(x) =`TopLeft`
- **Picture:** TempControl(x) & Loc(x) =`TopLeft`
- **Reduced KB:** TurnUp(p)& Actee(p) = x & TempControl(x)

The information Loc(x) =`TopLeft` can be removed from the information in the KB ('Turn up the temperature control, which is located on the top left of the machine') because this information is present in the picture. None of the constraints mentioned above is violated by this reduction.

5 Conclusion and Related Work

This paper has taken WYSIWYM into the area of *authoring text + pictures documents*. This is the first comprehensive exposition of *multimedia* WYSIWYM; specific aspects have been described more fully elsewhere. When compared with other approaches (see e.g., Maybury and Wahlster 1998), the following salient properties may be noted:

- Multimedia WYSIWYM involves the interactive specification of the document at a high level of abstraction (in the spirit of requirements 1-3 mentioned in the Introduction). This is unlike what is done in low-level authoring tools (e.g., *Word*) or even in Bateman et al. (2001), which appears to force authors to think in terms of the actual words in the document. Likewise, the author does not have to know what pictures the library contains. If more pictures become available later, the system will take these into account by re-generating the text and fitting in new pictures where this is advantageous.
- Subsequent versions of WYSIWYM are allowing authors to enter ever more detailed information. Unlike most systems, (e.g., Maybury 1993, André and Rist 1995), the interface does not require the author to understand a formal language, and using one uniform interface supports all actions that involve the editing of semantic representations, regardless of the modality of the presentation involved. At the same time, WYSIWYM avoids the pitfalls of open natural language interpretation. This is unlike the ALFresco system, for example, which has to interpret unconstrained natural language (Stock 1991).
- Unlike most systems (e.g., Kerpediev and Roth 2000, André and Rist 1995), WYSIWYM is not limited to diagrammatic representations, as we have seen. Without the benefit of WYSIWYM, indexation would be extremely burdensome (cf., Enser 1995), especially because our library contains semantic representations that are far more detailed than those in most current picture retrieval systems (e.g., Van de Waal 1995). Although we have not used WYSIWYM for large-scale annotation, our experience implementing ILLUSTRATE, and ap-

plying it to the pictures used by one company, suggests that it does make annotation easy: The interface does not only prevent users from making syntax errors or omitting crucial information; its T-Box also ensures that only relevant properties can enter an indexation.

- Pictures are included by a simple Information Retrieval algorithm. Standard retrieval methods typically have to be probabilistic because, in retrieval, mismatches between the search terms and the material in the search space (Van Rijsbergen 1985) tend to be frequent: two words may be nearly synonymous, somewhat synonymous, and so on. When search terms and indices are created by means of one and the same interface – as in the case of Multimedia WYSIWYM – such situations can only arise when the representation language (as specified in the T-box) is badly designed. This does not mean that the process by which pictures are retrieved (i.e., **Rule B**) is fault proof, and probabilistic methods may, for example, be used for choosing between two pictures, in a situation where each of the two lacks a different part of the Target Information, so that none of the two logically implies the other. (For discussion and criticism of our **Rule B**, see Pineda 2000.) Note that our method of annotating and searching for pictures can also be applied to the retrieval of pictures (or other objects whose meaning can be captured by Description Logic) *outside* document authoring: whenever pictures are to be retrieved, a WYSIWYM-style interface can be invoked to (1) annotate them (i.e., to create index terms) and (2) construct a query.[6] Note also that, in general-purpose retrieval, it is acceptable to retrieve more than one picture in response to a given query, leading to some fairly straightforward adaptations of **Rule B**.[7] Such an 'annotate & retrieve' approach to picture retrieval is consistent with ideas that have recently been put forward under the banner

[6] Compare Piwek et al. 2000, where applications of WYSIWYM to database query are described. (In their case, the objects retrieved are rules of *maritime law*, not pictures.)

[7] Compare a similar approach described in Schreiber et al. (2001), for example, who appear to retrieve *all* pictures whose annotations logically *imply* the search expression. Instead of WYSIWYM, Schreiber et al. use Protegé-2000, a RDFS-based ontology editor.

of the *semantic web* (Berners-Lee et al. 2001) and would become of great practical interest if larger picture collections were annotated than the ones we have so far experimented with.

- Unlike other systems that we are aware of, WYSIWYM allows a 'mixed initiative' (e.g. Gentner 1992) approach to multimedia document authoring. ILLUSTRATE implements a simple version of mixed initiative, by letting the author determine *what* to illustrate (by selecting, effectively, a section of the KB), and letting the system determine *how* to illustrate it. This approach could be tilted further towards system initiative: a RICHES-style automatic rule (section 3.3) could be used as a first approximation, allowing the author to removed or add pictures using WYSIWYM. It could also be tilted further towards user initiative, for example, by letting the system select several pictures, from which the user can make a choice.

An important limitation of present incarnations of multimedia WYSIWYM is the fact that they are unable to *generate* images. Although there is no obstacle *in principle* against applying WYSIWYM to text + diagrams (where the diagrams may be generated in the style of Kerpediev and Roth (2000), for example) we have not tried this out. (For other limitations, see section 4.1.)

Let us step back and ask what is essential about multimedia WYSIWYM. Many details of the systems described here could have been different. For example, the exact nature of the semantic representations employed is open for debate (see Brun et al. 2000 for a possible alternative). Which properties of multimedia WYSIWYM *matter*?

'High-level' (cf. section 1) authoring hinges on an ability to associate multimedia objects with *semantic representations*, and on an ability to manipulate these representations themselves (rather than their textual/pictorial instantiations only). When multimedia documents are authored, one representation language should be used for all media. In the case of an author requesting illustration of a particular document part, for example, the semantic representation enables the system

- to determine which picture is most suitable,

- to find a suitable location for it in the document,
- to adapt the text by omitting superfluous information from it, and
- to generate appropriate (document-deictic) cross-references to the text.

Our own implementation of this idea rests on two pillars: Description Logic (e.g., McGregor and Oates 1987) and WYSIWYM editing (Scott et al. 1998, Power and Scott 1998, Scott 1999). The former provides a multimedia interlingua, the latter a method for specifying and manipulating meaning representations.

APPENDIX: ILLUSTRATE at work

Here, we show in more detail how ILLUSTRATE[8] operates by elaborating on one of our examples. We show **(1)** how the content of a document is specified using a WYSIWYM interface, **(2)** how pictures can be annotated using the same tool, and **(3)** how the system selects an annotated picture to illustrate a designated part of the document. ILLUSTRATE is operational, but its language generation capabilities have been kept to a minimum, allowing us to focus on picture annotation and selection. (For WYSIWYM systems with more elaborate language generation capacity, see e.g., Power and Scott 1998, Scott et al. 1998, Scott 1999, Piwek 2000, Bouayad-Agha et al. 2002.) The generator is powerful enough to express most of the sentences in our example; information which the language generator cannot express is expressed by ILLUSTRATE using description logic formulas, which are displayed using labeled directed graphs (see below).

Suppose you are working for a pharmaceutical company. Together with a number of colleagues, you are responsible for composing *patient information leaflets* in a range of languages, and possibly in a range of styles. Your own preferred language is English, but your colleagues may prefer other languages. One colleague is responsible for maintaining a suitable collection of pictures, which may also be used in a

[8] ILLUSTRATE is written in Prolog and Lisp. Modules for knowledge management and language generation are implemented in Prolog; these are linked through a Unix socket to a user interface implemented in CLIM (1994).

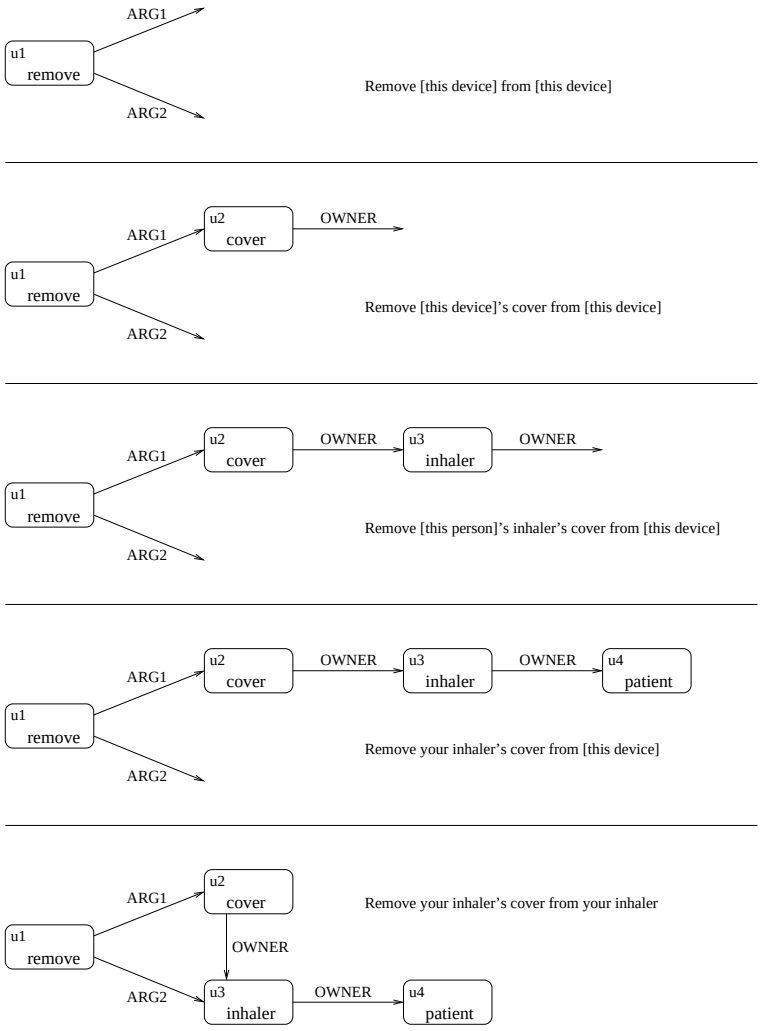

Fig. 5. Knowledge editing sequence

number of other leaflets; this collection may be updated regularly, in ways that you are not necessarily aware of. The WYSIWYM-assisted authoring process of a leaflet for an English-speaking readership proceeds as follows.

1. Building a KB to be expressed by text. You can build up an A-box that corresponds with the example in section 2 by means of the editing sequence shown in Fig.5. Each rectangle in the network diagrams represents an instance, labelled with an identifier (e.g., `u1`) and a type (`remove`). Attributes are represented by

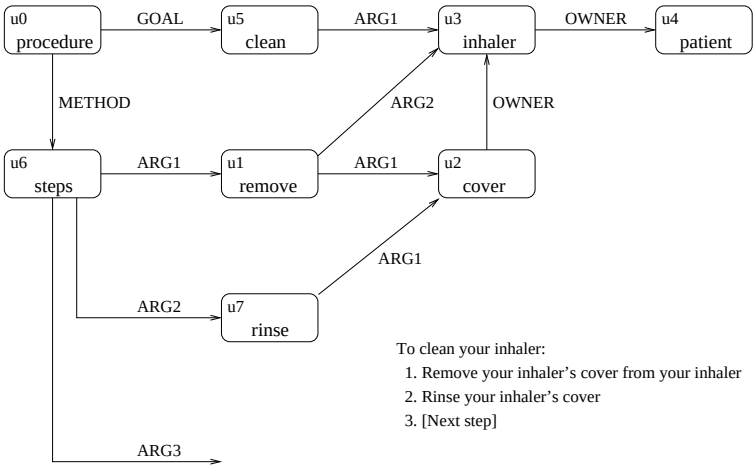

Fig. 6. Knowledge editing sequence

labelled arcs pointing to the value. The A-box is actually implemented by a set of Prolog assertions describing the network; for example, the A-box obtained by the end of the sequence would be represented by the following set of assertions:

```
remove(u1)
arg1(u1,u2)
arg2(u1,u3)
cover(u2)
owner(u2,u3)
inhaler(u3)
owner(u3,u4)
patient(u4)
```

This is equivalent to the more readable formula that we have encountered in section 3 (replacing some of the predicate names by more transparent synonyms and replacing one set of variables by another):

$$Remove(p)\ \&\ Source(p) = y\ \&\ Inhaler(y)\ \&\ Actee(p) = z\ \&$$
$$Cover(z)\ \&\ Owner(z) = y\ Owner(y) = u\ \&\ Patient(u).$$

After the very first stages of editing, the A-box will, of course, tend to be much larger than this formula; the removal of the cover might, for example, be the first of two steps that are required when an inhaler is cleaned, as in Fig.6. For current purposes, let us suppose that the network in Fig.6 represents the entire KB, which can be expressed in the feedback text displayed in the figure. Note that the text is almost excessively explicit, to facilitate further editing. It would be easy to generate

an output text like *'To clean your inhaler, first remove its cover then rinse it in water'*, but ILLUSTRATE is currently limited to the generation of feedback texts.

2. Building a library of annotated pictures. Suppose your colleague, the librarian, has a collection of pictures that are potentially relevant for your leaflet. In practice, such a collection will contain at least a few dozen pictures, but let us assume, for the sake of illustration, that there are only three. The librarian scans them, and saves them in `eps` format, whereafter he can start annotating them to allow the system to take their intended meaning into account. Each of the three annotations will consitute a tiny, independent A-box, each conforming with the T-box that also governs the A-box created in **(1)**. Let Fig.1 be the first of the three pictures, called `remove-cover-inhaler.eps`, and let it be annotated as explained in the body of the text. (In its present form, ILLUSTRATE supports feedback texts in English only, but the construction of generators in other languages along the lines of Bouayad-Agha (2002) would be unproblematic.) The editing process is exactly the same as that in Fig.5, except that it is not known whether the Actee of the Remove action is an inhaler or another kind of device (the assumption being that the picture itself does not allow one to decide). As a result, the system takes `remove-cover-inhaler.eps` to mean

`remove-cover-inhaler.eps`: $Remove(p)$ & $Source(p) = y$ & $Inhaler(y)$ & $Actee(p) = z$ & $Cover(z)$ & $Owner(z) = y$.

At this point, the library contains one picture. To allow us to illustrate the picture selection process, we will populate the library with two other pictures. The first one, `remove-cover-autohaler.eps`, is similar to the original one, but replacing the inhaler by an *auto*haler (this is a different but similar looking device):

`remove-cover-autohaler.eps`: $Remove(p)$ & $Source(p) = y$ & $Autohaler(y)$ & $Actee(p) = z$ & $Cover(z)$ & $Owner(z) = y$.

(For simplicity, we use the same variables as those in the formula representing `remove-cover-inhaler.eps`, even though the system uses different ones.) The last

picture, called `remove-cover-something`, depicts another removal action involving an inhaler, but the picture does not allow one to determine what it is that is being removed, causing the annotator to create the following annotation for it:

`remove-cover-something.eps`: $Remove(p)$ & $Source(p) = y$ & $DeviceorDevicepart(y)$ & $Actee(p) = z$ & $Cover(z)$ & $Owner(z) = y$.

Note that the choice of annotation is the annotator's responsibility: the system has no awareness of what the picture means or how it looks. On the positive side, this means that canned text or – format permitting – sound or video can be annotated in exactly the same way. On the negative side, it means that there is no guarantee that the annotation correctly reflects the meaning of the annotated material. Selecting the correct meaning from among all the ones allowed by the T-box is the responsibility of the annotator, who is therefore assumed to be a domain expert.

3. Requesting illustration; selection of illustration. Granted that a WYSIWYM-annotated library has been created by a 'librarian' as described under **(2)**, you as an author can request that a part of your doument be illustrated by the system. In principle, you might do this by interacting with the *output* text, but since generation of output texts is not implemented in ILLUSTRATE, we assume that you interact with *feedback* texts.[9] Suppose the author acts as depicted in Fig.2 (repeated here as Fig.7), by choosing the option *Add illustration* after selecting the sentence 'Remove your inhaler's cover from your inhaler'. (The other options, like 'Copy' and 'Global scope' are semantic operations on the KB, allowing the author to go on expanding or modifying it, as described under **(1)**.) As a result, ILLUSTRATE will look for pictures whose annotations suit the target information.

Target information: $Remove(p)$ & $Source(p) = y$ & $Inhaler(y)$ & $Actee(p) = z$ & $Cover(z)$ & $Owner(z) = y$ & $Owner(y) = u$ &$Patient(u)$.

As we have have seen under **(2)**, the library contains three candidate illustrations:

[9] To accommodate authors comfortable with Description Logic, illustration can also be requested by clicking on a part of the A-box graph, thereby bypassing the feedback as well as output text. If the author clicks on node `u1` of picture 5 or 6, for example, the effect is the same as when she clicks on the sentence 'Remove your inhaler's cover from your inhaler'.

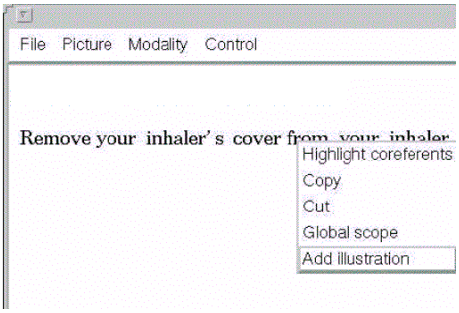

Fig. 7. Screendump: Author makes a requests for illustration

(a) `remove-cover-inhaler.eps`,
(b) `remove-cover-autohaler.eps`, and
(c) `remove-cover-something.eps`.

Following **Rule B**, picture (b) drops out, since it contains information not implied by the target information. Pictures (a) and (c) are both suitable, since both are logically implied by the target information (i.e., they do not contain unwanted information). Of these two, (a) is preferred because it is logically stronger than (c), containing more information about the `Source` of the removal. As a result, (a) is selected and integrated into the document.[10] Textual adaptations to the appearance of the picture may be made in principle, as explained in section 4.2. Given the limited content of the picture library exemplified in this Appendix, (the implemented version contains 35 annotated pictures), no picture would have been selected if the author had asked for illustration of the phrase *your inhaler*.

Note that this pictorial intermezzo has left the semantic content of the document as a whole unchanged. When this phase in the authoring process is concluded, the author (or one of his colleagues, for that matter) can complete the content of the A-box by further expanding the network displayed in Fig.6.

[10] As explained in section 3.4, ILLUSTRATE puts all pictures at the bottom of the page, while a more principled approach, which finds a place in the document near the relevant *target information*, was chosen in RICHES.

References

ABPI (1997). The Association of the British Pharmaceutical Industry, *1996-1997 Compendium of Patient Information Leaflets.*

AIR (1995). Special Issue, edited by P. Mc Kevitt, on Integration of Natural Language and Vision Processing: Intelligent Multimedia. Artificial Intelligence Review 9, Nos.2-3.

E. André and Th. Rist (1995). Generating Coherent Presentations Employing Textual and Visual Material. *Artificial Intelligence Review* **9**:147-165.

D. Barker-Plummer and M. Greeves (1995). Architectures for Heterogeneous Reasoning. In J.Lee (Ed.) *Proc. of First International Workshop on Intelligence and Multimodality in Multimedia Interfaces: Research and Applications* (IMMI-1), Edinburgh.

J. Bateman, J. Delin, and P. Allen (2000). Constraints on layout in multimodal document generation. In Procs. of First Int. Conf. on Natural Language Generation, workshop on Coherence in Generated Multimedia. Mitzpe Ramon.

J. Bateman, Th. Kamps, J. Kleinz, and K. Reichenberger (2001). Towards Constructive Text, Diagram, and Layout Generation for Information Presentation. *Computational Linguistics* 27, 3: 409-449.

M. Bordegoni, G. Faconti, S. Feiner, M.T. Maybury, T. Rist, S. Ruggieri, P. Trahanias, and M. Wilson (1997). A Standard Reference Model for Intelligent Multimedia Presentation Systems. Computer Standards & Interfaces 18, pp. 477-496.

Bouayad-Agha, N., R. Power, D. Scott and A. Belz (2002). PILLS: Multilingual generation of medical information documents with overlapping content. In Proceedings of LREC 2002, pp. 2111-2114.

Brun et al. (2000). C.Brun, M.Dymetman, and V.Lux. Document Structure and Multilingual Authoring. In Proc. of First Int. Conf. on Natural Language Generation. Mitzpe Ramon.

C. Cahill et al. (2000). Cahill, L., Carroll, J., Evans, R., Paiva, D., Power, R., Scott, D.R. & van Deemter, K. (2001). From RAGS to RICHES: exploiting the potential of a flexible generation architecture. Proceedings of ACL/EACL 2001, pp. 98-105.

CLIM (1994). Common Lisp Interface Manager: User Guide. Franz Inc.

R. Dale and E. Reiter (1995). Computational Interpretations of the Gricean Maxims in the Generation of Referring Expressions. *Cognitive Science* 18: 233-263.

P. Enser (1995). Progress in Documentation; Pictorial Information Retrieval. *Journal of Documentation*, Vol.51, No.2, pp.126-170.

G. Erbach (1995). ProFIT: Prolog with Features, Inheritance and Templates. Seventh Conference of the European Chapter of the Association for Computational Linguistics, Dublin, pp. 180-187.

ETAI (1997, 1998). ETAI News Journal on Intelligent User Interfaces, Vol 1, No's 1 and 2.

S.K. Feiner and K.R. McKeown (1998). Automating the Generation of Coordinated Multimedia Explanations. In Maybury and Wahlster (1998)

D. Gentner (1992). Interfaces for Learning: Motivation and the Locus of Control. In F.L.Engel, D.G. Bouwhuis, T.Boesser, and G.d'Ydewalle (Eds.), *Cognitive Modelling and Interactive Environments in Language Learning.* NATO ASI series Vol. F87. Berlin. Springer.

W.H. Graf, S. Neurohr, and R. Goebel (1996). A Constraint-Based Tool for the Pagination of Yellow-Page Directories. In U. Geske and H. Simonis (Eds.) Procs. of KI96 workshop on declarative constraint programming. GMD-Studien 297, St. Augustin.

H. Kamp and U. Reyle (1993). *From Discourse to Logic.* Kluwer Academic Publishers, Dordrecht.

Kerpediev and Roth (2000). Mapping Communicative Goals into Conceptual Tasks to generate Graphics in Discourse. In Procs. of Intelligent User Interfaces (IUI-00), New Orleans.

H.J. Levesque (1986). Making Believers out of Computers. Artificial Intelligence 30, pp.81-108

J. Masthoff (2000). Analytical Usability Evaluations of WYSIWYM. ITRI internal report (`http://www.itri.bton.ac.uk/research.html#WYSIWYM`).

M. Maybury (1993). Planning Multimedia Explanations Using Communicative Acts. In Procs. of AAAI Workshop on Intelligent Multimedia Interfaces; reprinted in Maybury and Wahlster (Eds.) 1998.

M. Maybury and W. Wahlster (1998). Readings in Intelligent User Interfaces. Morgan Kaufmann, San Francisco.

R. McGregor and R. Bates (1987). The LOOM Knowledge Representation Language. In Procs. of the Knowledge-Based Systems Workshop, *St. Louis, April 21–23.*

Mittal et al. (1998). Mittal, Moore, Carenini, and Roth. Describing Complex Charts in Natural Language: A Caption Generation System. *Comp. Ling.*, **24**, No.1.

I. Paraboni and K. van Deemter (2002). Towards the Generation of Document-Deictic References. In K.van Deemter and R.Kibble (Eds.), *Information Sharing: Reference and Presupposition in Language Generation and Interpretation*, CSLI Publications, Stanford.

L. Pineda (2000). Spatial Language, deixis and illustration. In Procs. of workshop "Coherence in Generated Multimedia", associated with First Int. Conf. on Natural Language Generation. Mitzpe Ramon, Israel.

P. Piwek (2000). A Formal Semantics for Editing and Generating Plurals. In Proceedings of COLING 2000, Saarbruecken, Germany.

P. Piwek et al. (2000) P.Piwek, R.Evans, L. Cahill and N. Tipper. Natural Language Generation in the MILE System. In Proceedings of the 'IMPACTS in NLG' Workshop, Schloss Dagstuhl, Germany, July 2000.

R. Power and D. Scott (1998). Multilingual Authoring using Feedback Texts. In *Proc. of COLING/ACL conference*, Montreal.

S. Roth and W. Hefley (1993). Intelligent Multimedia Presentation Systems: Research and Principles. In M.Maybury (Ed.) *Intelligent Multimedia Interfaces*, AAAI Press, pp.13-58.

A. Schreiber, B. Dubbeldam, J. Wielemaker, and Bob Wielenga. Ontology-Based Photo Annotation. *IEEE Intelligent Systems*, May/June 2001.

D. Scott, R. Power, and R. Evans (1998). "Generation as a Solution to its own Problem", Accepted for Proc. of 9th International Workshop on Natural Language Generation.

D. Scott (1999). The Multilingual Generation Game: authoring fluent texts in unfamiliar languages. Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99).

O. Stock (1991). Natural Language and Exploration of an Information Space: the ALFfresco Interactive System. In M. Maybury and W. Wahlster (1998).

K. van Deemter (1998). Retrieving Pictures for Document Generation. In: Proc. of 14th Twente Workshop on Language Technology, on Multimedia Information Retrieval (TWLT14), University of Twente, The Netherlands (1998)

K. van Deemter (1999). Document Generation and Picture Retrieval. In Proc. of Third Int. Conf. on Visual Information Systems, Amsterdam, Springer Lecture Notes.

K. van Deemter and R. Power (1999). Inclusion of Picture Sequences in Generated Documents. In Proc. of fourth Portuguese Conf. on Artificial Intelligence, Evora, Springer Lecture Notes.

K. van Deemter and R. Power (2000). Authoring Multimedia Documents using WYSIWYM Editing. In Procs. of COLING conference, Saarbruecken.

H. van de Waal (1995). ICONCLASS; *An iconographic classification system.* Amsterdam 1973-1985 (17 vols). ISBN 0-7204-8264-X. See also
<http://iconclass.let.ruu.nl/home.html>.

C.J. van Rijsbergen (1989). Towards an information logic. In: Proc. ACM SIGIR.

C. Venters and M. Cooper (2000). A Review of Content-Based Image Retrieval Systems. `http://www.jtap.ac.uk/reports/htm/jtap-054.html`

W. Wahlster, E. André, W. Finkler, H.-J. Profitlich, and Th.Rist (1993). Plan-based Integration of Natural Language and Graphics Generation. *Artificial Intelligence* 63, p.387-427.

M.X. Zhou and S.K. Feiner (2001). Improvise: Automated Generation of Animated Graphics for Coordinated Multimedia Presentations. In H.Bunt and R.-J. (Eds.) *Cooperative Multimodal Communication.* Springer, Berlin.

T. Berners-Lee, J. Hendler, and O. Lassila. A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*, May 2001.