Document Content Authoring and Hybrid Knowledge Bases

Marc Dymetman
Xerox Research Centre Europe
6 chemin de Maupertuis
38240 Meylan
email: marc.dymetman@xrce.xerox.com

Abstract

We present a principled approach to the problem of connecting a controlled document authoring system with a knowledge base. We start by describing *closed-world authoring* situations, in which the knowledge base is used for constraining the possible documents and orienting the user's selections. Then we move to *open-world authoring* situations in which, additionally, choices made during authoring are echoed back to the knowledge base. In this way the information implicitly encoded in a document becomes explicit in the knowledge base and can be re-exploited for simplifying the authoring of new documents. We show how a Datalog KB is adequate for the closed-world situation, while a Description Logic KB is better-adapted to the more complex open-world situation. We stress the strong connections between the later case and current work on hybrid knowlege bases of the Datalog-DL type. All along, we pay special attention to logically sound solutions and to decidability issues in the different processes.

1 Introduction

Recently there has been a surge of interest in *interactive natural language generation systems* [12, 14, 4, 16]; such systems rely on a capability of generating a natural language text from an abstract content representation, but — contrary to traditional NLG systems — this representation is only partially available at the beginning of the text production process; it is then gradually completed by a human author, typically using content-selection menus correlated with regions of the evolving generated text..

One such system, MDA (Multilingual Document Authoring) [8, 2] is based on a formal specification — using a variant of Definite Clause Grammars (DCGs) [13] — of what counts as a valid abstract content representation. The different derivation trees in the grammar correspond to texts with different contents, and at each step of the authoring process the user is asked to make interactive choices on how to expand the current partial derivation tree one step further. There are important analogies between this process and the process of authoring an XML document under the control of a DTD or a Schema, but DCGs are more expressive in terms of the contextual constraints that can be expressed and also are more adapted to the production of grammatical text.[1]

[1]The grammars used in MDA are typically more "semantically" than "syntactically" oriented, and a choice

In published MDA work, all the knowledge about what constitutes a valid document is provided in the grammars, with no clear separation between (1) world knowledge (the fact that a certain pharmaceutical drug contains some molecule makes it dangerous for a certain patient condition) and (2) constraints about document organization (if a certain drug is dangerous for a certain condition, then a warning should be generated at a certain place in the document).

A more principled and modular solution is to leave in the grammar all constraints pertaining to document/textual organization, and to use an external logical theory to express knowledge about the world described by the documents. A document will then be constrained to have a semantic interpretation that is compatible with the external theory.

In this paper we will:

- Provide a *formally precise and computationally tractable model* for this approach. The logical theory we will be using will take the form of a Description Logic (DL) knowledge base [6]; Such representations are currently given a lot of attention in the knowledge representation community and in activities around the Semantic Web, and have recently started to attract attention in the computational linguistics community as well [9, 17];

- Show how this model can be used not only for constraining the document during the authoring process, but also to use the document as a source of new knowledge to be added in a logically sound way to the KB (*knowledge acquisition*);

- Discuss conditions under which the whole process of authoring is *decidable*.

The paper is organized as follows. We first describe a class of situations, *closed-world authoring*, in which the flow of information is strictly from the knowledge base to the document. The MDA approach is briefly presented, and we show how the document specification can be interfaced with an "informationally complete" KB, using a Datalog representation [3]; then we present conditions on the specification which guarantee decidability of the closed-world authoring process, that is, that guarantee that at each authoring step, the selections presented to the author are "real choices" which will not result in dead-ends at a later stage of authoring. We then move on to *open-world authoring*, in which the flow of information is bi-directional between the KB and the document. Now we start working with an "informationally incomplete" KB, using a Description Logic representation, which can be satisfied in several "possible worlds"; the document being authored has to be compatible with at least one of these possible worlds. We give conditions on the grammar which guarantee that, as long as the DL on which the KB is built is intrinsically decidable, then the authoring process as a whole is also decidable. We stress the strong connections of our approach with current work on hybrid knowlege bases of the Datalog-DL type [11, 7]. Finally we introduce a notion of *light semantics*, which corresponds to a restricted form of semantic interpretation for the document allowing exchange of information between the document and the knowledge base and permitting *knowledge acquisition* during the authoring process. In particular the knowledge gained during the authoring of a document can be re-used for simplifying the authoring of other documents.

---

between two alternatives for expanding a nonterminal in the grammar tends to correlate with a clear distinction of meaning in the final text. A given grammar covers a semantically unified class of documents (e.g. employment offers, drug package leaflets, etc.), in a way analogous to the customized XML DTDs used for technical documentation.

2 Closed-world authoring

MDA. We start by introducing briefly the MDA framework through a simplified example. The focus of this paper is on the document content aspects (as represented by what we call the abstract content tree) and not on the textual realization aspects, which are handled in a simplistic way here (see [8, 2] for details on MDA).

Grammar G1:

```
dfa1:     dfa(D,F,A) → "the drug", drug(D), "has the form",
                       dform(D,F), "and is administered by",
                       dadm(D,A).
dform1:   dform(D,F) → form(F), & df(D,F).
dadm1:    dadm(D,A) → admin(A), comments(D,A).
dadm1:    dadm(D,A) → admin(A), comments(D,A).
coms1:    comments(D,A) → " ", & da(D,A).
coms2:    comments(D,A) → comments(D,A),
                          ";", comment(D,A).
com1:     comment(D,A) → "strictly follow instructions".
com2:     comment(diprox,A) → "take a glass of water".
diprox:   drug(diprox) → "Diprox".
xenor:    drug(xenor) → "Xenor".
burpal:   drug(burpal) → "Burpal".
tablet:   form(tablet) → "tablet".
solution: form(solution) → "solution".
swallow:  admin(swallow) → "swallowing".
chew:     admin(chew) → "chewing".
drink:    admin(drink) → "drinking".
```

Auxiliary clauses D1:

```
df(diprox,tablet).
df(xenor,tablet).
df(burpal,solution).
da(diprox,swallow).
da(xenor,chew).
da(burpal,drink).
```

The form of grammar G1 is a variant of the DCG format [13]: (1) each of the grammar clauses is given a unique name (e.g. dfa1); (2) the nonterminals are notated in lowercase and are parameterized by variable or ground terms; (3) the terminals are enclosed in double quotes; (4) the auxiliary predicates (a.k.a. Prolog calls, usually enclosed in curly brackets) appear after the ampersand sign.

Free generation. If we start from the initial nonterminal dfa(D,F,A) and expand it nondeterministically until we get to terminal strings (so-called free generation mode), we can obtain (among others) the texts:

(T1) "*the drug Diprox has the form tablet and is administered by swallowing*",

(T2)*"the drug Xenor has the form tablet and is administered by chewing; strictly follow instructions"*,

but not the text:

"the drug Burpal has the form tablet and is administered by swallowing".

Authoring. The authoring mode is different from the free generation mode in that it gives the author the responsibility of choosing expansions for nonterminals rather than enumerating all possible expansions nondeterministically. Thus, after all the obligatory expansions from dfa(D,F,A) (expansions for which there is only one possibility in the grammar) have been done, the frontier of the derivation tree contains some terminals and the nonterminals drug(D), form(F), admin(A), comments(D,A), and has to satisfy the constraint df(D,F).[2] At this point the user can freely choose which of these nonterminals to expand next — say form(F). There are two possible ways to expand this nonterminal: through the clause of name tablet or through the clause with name solution, and the system displays to the user a menu listing these two choices. Assume that the author chooses tablet. The nonterminal form(F) is expanded into the terminal "tablet", F is unified with tablet, and the process is repeated until no more nonterminal needs to be expanded.

At the end of this process, the collection of choices that the user has made can be represented as a tree labeled by names of clauses, for instance:

(AT1) dfa1(diprox, dform1(tablet), dadm1(swallow,coms1))

from which a complete derivation tree can be reconstructed as well as the associated terminal string, which in this case is seen to be equal to T1.

Such a tree of choices as AT1 will be called an *abstract content tree*, or simply an *abstract tree*. Different abstract trees correspond to different sets of choices of content and also to different document instances in the class of documents associated with the grammar. It is then natural to see an abstract tree as a representation of the content of a document relative to this class of documents.[3]

Life/death issues There is one important issue that we did not discuss in the explanation just given, namely how exactly the system determines which choices to propose the user once he has selected a new nonterminal to be expanded. One possibility is to present him with all the possible names of clauses which are headed by the nonterminal in question (as was done for form(F)), but then it is possible that the author makes a choice that will never lead to a complete valid document.

For instance, let us go back to the point just after the author has chosen tablet as the clause for expanding form(F); at this point the nonterminals on the frontier of the derivation tree are: drug(D), admin(A), comments(D,A), with the constraint df(D,tablet) in the background. Suppose the author next chooses to expand admin(A); if the system was working in a naive fashion, it would then display the choices swallow, chew, and drink. However it is easy to see that drink is in fact ruled out as a choice: any complete document would eventually have to

[2] What we call frontier here is sometimes called sentential form when speaking about context-free grammars.

[3] This approach to document content stems from the work of Aarne Ranta on his "Grammatical Framework (GF)" [15, 16] in which he was inspired by the interactive proof editors in a higher-order typed/functional setting such as ALF and COQ in which the user attempts to build a proof of a formula through stepwise top-down refinements of a partial proof. In the present paper the abstract trees can be seen as proofs of an initial goal in a logic programming setting.

satisfy the constraints df(D,tablet) and da(D,drink), but there is no drug in the database which is compatible with both this form and this administration. We can say that drink is a "dead" choice in this context.

In order to prevent the author from entering a dead-end, what is really needed is for the system to foresee such possible clashes and to present to the author only those choices which may eventually lead to a valid document; in the case at hand, it should present the "live" choices swallow and chew.

Remark. When exactly one choice is possible, the system should not even present any choice to the author, but make the only possible expansion decision on its own: authoring should be done automatically at that point. In these cases the authoring mode becomes closer to the classical non-interactive NLG mode, and in the limit, when knowledge-base inferences force all authoring choices, the two modes converge.

Finitely-parameterized grammars, Datalog, and decidability of life/death In the current MDA system, the method for determining whether a choice is live or dead is incomplete. This is due to the fact that the nonterminal parameters can be terms of arbitrary complexity (built from variables, constants *and functional symbols*) and then it is easy to simulate with a DCG an arbitrary Prolog program.[4] Determining whether the initial nonterminal may lead to a complete valid document is then undecidable in general. It is usually possible for the grammar writer to exercise some care in designing the grammars so that life/death problems do not hinder the authoring process in practice, but a principled solution would be preferable.

In order to tackle this problem, we will be making two fundamental assumptions: (i) the nonterminal parameters in the grammar clauses — as well as the goal arguments in the auxiliary program clauses — are variables or constants; (ii) all variables take their value in the finite set of constants present in the grammar and auxiliary clauses .

Under these assumptions, we are now dealing with a DCG with *finite-domain parameters* both for its grammar and for its auxiliary predicates components. The auxiliary predicate component is then formally the same as a Datalog database [3], as in our example D1.[5]

We can then see the authoring model as consisting of two components: a finitely parameterized DCG, and a Datalog database.

Now, it is striking that, when working with finite-domain DCGs, not only the auxiliary predicate component, *but also the grammar component*, has formal similarities to a Datalog base: in fact, if one "forgets" in the grammar G1 all the terminal strings, then one obtains a Datalog program DP1:

DP1:

dfa1: dfa(D,F,A) ← drug(D), dform(D,F), dadm(D,A).
dform1: dform(D,F) ← form(F), & df(D,F).
dadm1: dadm(D,A) ← admin(A), comments(D,A).
coms1: comments(D,A) ← & da(D,A).
coms2: comments(D,A) ← comments(D,A), comment(D,A).
com1: comment(D,A).

[4] Even without the use of auxiliary predicates: a pure Prolog program is equivalent to a DCG generating empty strings.

[5] The database D1 only contains facts (Datalog's EDB), but it could also contain recursively defined predicates (Datalog's IDB) without impact on the discussion.

com2: comment(diprox,A).
diprox: drug(diprox).
xenor: drug(xenor).
burpal: drug(burpal).
tablet: form(tablet).
solution: form(solution).
swallow: admin(swallow).
chew: admin(chew).
drink: admin(drink).

Deciding the productivity of a parameterized nonterminal in the combination G1+D1 is then *formally equivalent to proving it as a program goal* in the combination DP1+D1 (which is itself a global Datalog program), and a derivation in G1 has a one-to-one correspondence to a proof in DP1.

For instance, deciding the productivity of the nonterminal dfa(D,tablet,drink)is equivalent to proving the goal dfa(D,tablet,drink) in the Datalog program DP1+D1: because no such proof can be found, the nonterminal is not productive.

Now, the interest of this translation is that provability of a goal in a Datalog program is not only known to be decidable, but also to be amenable to efficient implementation [1].

Consider the situation discussed before, just after the author has chosen the form tablet, and at the point where the system needs to present him with a list of choices for admin(A). At that point, the system is confronted with the following question: what are the possible values for A such that the following goal:

drug(D), admin(A), comments(D,A), df(D,tablet)

is satisfiable?

This question can be succinctly represented as the following conjunctive Datalog query:

answer(A) ← drug(D), admin(A), comments(D,A), df(D,tablet)

for which a number of optimization techniques exist (see [3, 1]), and which returns as possible values for A the set {swallow, chew}.[6] The advantage for authoring is clear: at each choice point, the system is capable to return a valid list of choices more efficiently than by applying more naive techniques. It is also worthy of note that some *fundamental issues in authoring are so closely connected with database query optimization*.[7] [8]

[6] In this case, the set of possible values for the parameter A coincides with the set of possible values for the names of the expanding clauses for admin(A). In general it is not the case, but it is simple to add a parameter to each nonterminal that indexes its (finitely many) possible expanding clauses.

[7] A DCG is nothing else than a context-free grammar with parameterized nonterminals and a unification mechanism between the parameters. Because of the analogy between DTD/Schemas and CFGs, it seems likely that the same approach could be useful for extending XML-based authoring through the use of finite-domain parameters and unification.

[8] The fact that the program DP1 is equivalent to G1 *as far as nonterminal productivity is concerned* does not mean that the two objects are equivalent for authoring purposes. The grammar associates different texts with different derivations of the *same* ground nonterminal (for instance, there are an infinite number of texts produced by comments(diprox,tablet), corresponding to different combinations of coms1, coms2, com1, com2.), whereas the program is of interest to us here not in the different proofs of a given ground goal, but in the fact that this goal is provable or not. Note that the clause of name coms2 can be eliminated from the program DP1 without changing its interpretation (because in order to prove comments(D,A) it requires a proof of comments(D,A)), but making the program non-recursive and therefore simplifying the check for productivity; eliminating the same clause from G1 would however completely change the meaning of the grammar.

3 Open-world authoring

In an authoring context, some grammatically valid documents will never be authored because they do not correspond to any possible state of affairs. Typically the grammar specifies a much larger set of documents than the ones which are actually possible. If this were not the case, then an author would not have to take the trouble to direct the production process by making content choices that he alone can make. That is to say, a document which has actually been authored conveys more meaning than just stating "I am a valid document relative to the specification". However, in a closed-world environment as we have been discussing until now, that additional meaning has no explicit counterpart in the knowledge-base; it is only represented implicitly in the abstract content tree, in a form which is not perspicuous and would be difficult to re-use for the authoring of other documents or to share with other processes.[9]

In a closed-world context, the KB constraints which are tested during the authoring process are completely passive: they are seen purely as validity checks against the knowledge base.

By contrast, open-world authoring sees the KB constraints not only as checks, but also as conditions on the world being described. When authoring a document, the author is not neutrally picking out one of the documents valid relative to the KB, but asserting that the constraints *do* hold of the actual world.

Let us illustrate this idea. We are now viewing the formal specification of valid documents as consisting, as before, of a grammar of the type previously described (we will take again the grammar G1), but instead of a Datalog database, we are now using an *informationally incomplete* description logic knowledge base KB1:

KB1:

TBOX:
TabletDrugs = $\exists$df.{tablet}
SolutionDrugs = $\exists$df.{solution}
SwallowDrugs = $\exists$da.{swallow}
ChewDrugs = $\exists$da.{chew}
DrinkDrugs = $\exists$da.{drink}

Drugs = TabletDrugs $\uplus$ SolutionDrugs
Drugs = SwallowDrugs $\uplus$ ChewDrugs $\uplus$ DrinkDrugs

TabletDrugs = SwallowDrugs $\uplus$ ChewDrugs
SolutionDrugs = DrinkDrugs

ABOX :
df(burpal,solution)
da(burpal,drink)

This knowledge-base is written using a certain number of DL constructors — existential quantification, concept enumeration, disjoint union (an abbreviation: A = B$\uplus$C can be

[9] Note an analogy here with the Semantic Web perspective: tags used in XML documents may convey implicit semantic information, but in order to make this information sharable, it had better be represented explicitly in some formal knowledge representation.

replaced by the two constraints A=B⊔C and B⊓C = ⊥, and B⊎C⊎D is an abbreviation for (B⊎C)⊎D)[10] —, and we are assuming the unique name convention (all named individuals are different). The constructors which are used place the knowledge base in the class $\mathcal{ALCO}$ [6].

The TBOX can be glossed in the following way. The TabletDrugs are those drugs D for which df(D,tablet), the SolutionDrugs those drugs for which df(D,solution), ..., the DrinkDrugs those drugs for which da(D,drink). The drugs can come in either one of the two forms: tablet and solution, and in either one of the three administrations swallow, chew and drink. Finally TabletDrugs are either swallow drugs or chew drugs, whereas SolutionDrugs are always drink drugs. The ABOX says what we already know about the form and administration of Burpal.

The list of relations in D1 is compatible with KB1: indeed it is easy to see that one can obtain a model of KB1 by taking the relations of D1 along with the facts:

diprox: TabletDrugs
xenor : TabletDrugs
burpal : SolutionDrugs
diprox: SwallowDrugs
xenor : ChewDrugs
burpal : DrinkDrugs

In a certain sense the TBOX of KB1 can be seen as a conceptual schema for the database D1, which states certain general relations about the forms and administrations of drugs, or about the uniqueness of form and administration for a drug, but which does not say how many drugs there are or what are the properties of these drugs.

Valid abstract trees and incomplete KBs Let us return to our authoring example in this new context. We now associate grammar G1 with KB1 instead of DB1. We then make the assumption that all constant parameters appearing in the grammar (diprox, xenor, burpal, tablet, etc.) are to be considered distinct named individuals for the KB, and that the constraint relations (da, df) are all unary or binary and correspond to concepts or roles in the KB.

Let's now look again at the abstract tree AT1:

dfa1(diprox, dform1(tablet), dadm1(swallow,coms1))

This abstract tree is valid *relative to G1* (it corresponds to a possible complete derivation) but it is not necessarily valid relative to the combination <G1,KB1>; this notion is defined in the following way: because the abstract tree uniquely determines the set of rules which have been used for building the derivation, it also uniquely determines a set of associated KB constraints; thus AT1 is associated with the set of constraints: {df(diprox,tablet), da(diprox,swallow)}.

Now we say that AT1 is valid relative to the combination <G1,KB1> if and only if it is both valid relative to G1 and if its associated set of constraints is compatible with KB1. In other words we need to show that *the addition of the two constraints df(diprox,tablet), da(diprox,swallow) to the ABOX still leads to a satisfiable knowledge base*. This can be shown by exhibiting a model as we did a few paragraphs ago, and therefore AT1 is a valid abstract tree relative to <G1,KB1>.

[10] We see disjoint union as purely an abbreviation mechanism here, that is, we suppose that it is actually replaced through the expansions just defined. An actual constructor for disjoint union could however easily be introduced in the DL formalism itself.

The informal reasoning by which we just showed the satisfiability of KB1 extended with the two relations can also be established by a computational proof, due to the decidability of KB-consistency checking in $\mathcal{ALCO}$ [5, 10].

Open- vs. closed-world authoring, satisfiability vs. deducibility Note that validity of an abstract tree in the *open-world authoring* context involves the *satisfiability* of a conjunction of constraints relative to the knowledge base, whereas the notion of validity of an abstract tree in the *closed-world authoring* context involves the dual notion of *deducibility* of a conjunction of constraints relative to the knowledge-base (in the Datalog context, being true in the minimal Herbrand model is the same as being deducible from the Horn clauses constituting the base).

Decidability of the authoring process In order to illustrate the process, let's go back to the point in the authoring after all obligatory expansions of dfa(D,F,A) have been made, where the frontier of the derivation tree is drug(D), form(F), admin(A), comments(D,A), and where the user has chosen to expand form(F). There are apparently two possible expansions: the clauses with names tablet and solution. Before presenting these choices to the user, the system must check that they are live, namely, as before, that they may lead to a complete valid document.

Choosing the tablet expansion leads to the derivation frontier drug(D), admin(A), comments(D,A) with constraint df(D,tablet). In order to decide whether the frontier is live, the system needs to enumerate possible complete derivations of this frontier until it finds one that is satisfiable relative to KB1 and then return a positive answer, and if it does not find one, it should return a negative answer. In principle, the enumeration could never stop, but because of the finite parameter condition on the grammar, the system has only to enumerate a finite number of trees; this is because if a derivation tree is of the form S(... A1(... A2(...) ...) ...) where S is a ground instantiation of the initial nonterminal and A1 and A2 are the *same* ground instantiation of a nonterminal ("repetitive derivation"), then the satisfiability of S(... A1(... A2(...) ...) ...) relative to KB1 implies the satisfiability of S(... A2(...) ...): a model of the larger derivation tree is again a model of the smaller derivation tree. *This means that when checking life/death we do not ever need to consider a repetitive derivation during the enumeration of derivations.* In particular, because we are dealing with a finite parameter domain, the derivations that we need to consider have a bounded depth (otherwise we would necessarily encounter repetitive situations), and the decidability of the process follows.[11]

In the case of choosing tablet, the abstract tree AT1 is enumerated at some point in the process, and its satisfiability relative to KB1 can be decidably checked: tablet is then shown to be a live authoring choice. The same process shows solution to be live.

Now, let's go to the point where, after having chosen tablet, the author decides to select an expansion for admin(A). The derivation frontier is then drug(D), admin(A), comments(D,A), with the constraint df(D,tablet), and the apparently possible expansions are swallow, chew, and drink. Both swallow and chew can be seen to be live by a similar reasoning as before. In the case of drink, we have to check whether the sequence drug(D), comments(D,drink), with the constraint df(D,tablet) is live. Let's choose to expand comments(D,drink) first. The expansion coms2 leads to a repetitive situation (comments(D,drink) is above comments(D,drink) in the derivation path.) and is therefore discarded; the expansion coms1 leads to the frontier

[11] The same reasoning could be made for proving decidability in the closed-world case, instead of appealing there to the decidability of Datalog queries.

drug(D), with the constraints df(D,tablet) and da(D,drink). However the two constraints cannot be simultaneously satisfied in KB1; This can be shown by using the satisfiability check in KB1, but also by the following informal reasoning: df(D,tablet) and da(D,drink) imply that D is both in TabletDrugs and in DrinkDrugs; by the second fact it is in SolutionDrugs, but SolutionDrugs and TabletDrugs have an empty intersection. Thus all expansions of comments lead to invalidity; hence drink is not a live choice.

Open-World authoring and hybrid knowledge bases The process that we have just described for finding live selection, although decidable, is clearly not optimized. In the case of closed-world authoring that we discussed at the beginning of this paper, we said that, from the point of view of detecting life/death situations, a Datalog program such as DP1 could be used in place of the grammar G1, and that the combination of DP1 + D1 could be treated as a global Datalog program to which standard database optimization techniques could be applied. Is there some comparable possibility here? We do not have an answer at this point, but we will show that there are strong connections between our problem and problems considered in the field of hybrid Datalog-DL knowledge bases, and that this might provide directions in which to search for optimization techniques.

In the open-world authoring context that we have been discussing, all life/death issues can be stated relatively to the combination DP1 + KB1 instead of relatively to the combination G1 + KB1. Formally, we are now looking for proofs of the predicates of DP1 which are conditioned on the "acceptability" of combination of constraints relative to the knowledge base KB1, where we mean here by "acceptability" the *satisfiability* of this combination of constraints relative to KB, whereas in the case that we discussed before of the global Datalog program DP1 + D1, we were concerned with a notion of acceptability as *provability* relative to the database.

As an illustration, the abstract tree `dfa1(diprox, dform1(tablet), dadm1(swallow, comments1))` can be seen as a proof of the DP1 predicate `dfa(D,F,A)` which is conditional on the satisfiability of the constraints `df(diprox,tablet)` and `da(diprox,swallow)` relative to KB1.

Now, interestingly, the situation in which a combination of a Datalog program and a Description Logic knowledge base is considered is strongly reminiscent of work done in the field of hybrid Datalog-DL knowledge bases [11, 7], however we will see that our approach and these approaches are kinds of dual to each other.

The hybrid Datalog-DL knowledge bases consider the following problem: if P is a grounded instance of a predicate of the Datalog component DP of an hybrid knowledge base <DP,KB>, where KB is the description logic component, then P is considered to be acceptable (read provable) relative to <DP,KB> if and only if, *for any model* of the KB, there exists a proof of P in the program DP which is conditional on a set of constraints that are true in the model. A formal proof of the fact that P is acceptable takes the form of exhibiting a set of proof trees relative to DP such that the *disjunction* of the constraint combinations associated with each proof tree is *deductible* from the knowledge base (indeed, if such is the case, then in any model of KB, at least one of the proof trees will be conditioned on a combination of constraints true in this model). According to this approach, then, the goal `dfa(burpal,solution,drink)` is provable relative to <DP1,KB1>, but the goal `dfa(diprox,tablet,swallow)` is not.

According to our definition, however P is considered acceptable (read satisfiable) relative to <DP,KB> if and only if, *there exists a model* of the KB such that there exists a

proof of P in the program DP which is conditional on a set of constraints that are true in the model. A formal proof of the acceptability of P takes the form of exhibiting *one* proof tree relative to DP such as its associated combination of constraints is *satisfiable* relative to the KB. According to that approach, each of the goals `dfa(burpal,solution,drink)` and `dfa(diprox,tablet,swallow)` is satisfiable relative to <DP1,KB1>.

The acceptability problem in our approach leads to more local proofs than the one in the hybrid KB approach (which involve the simultaneous consideration of several proof trees) and therefore seems simpler. Still, our open-authoring approach and the hybrid KB approach, while different, have obvious affinities, and it might be hoped that some optimization results for the hybrid approach could be transferred to our case.

Light semantics and knowledge acquisition Let's step back and reconsider the rationale behind open-world authoring. We are considering a situation in which there is an "actual world" which is not completely known either to the knowledge base or to the author; however both the KB and the author are supposed to have correct partial knowledge about that world.

The system presents the author with a collection of documents which, *from its point of view*, are compatible with what *it* knows about the actual world. Among these documents, the author picks (during the authoring process) one document that, *from his point of view*, is compatible with what *he* knows about the actual world.

So the author is not passively exploring the space of document considered possible by the system (although that could certainly be a nonstandard mode of operation if the author takes a developer's hat and wants to see what the system believes is possible), but is actively committing to certain facts about the world.

What are these facts? What the author is producing is an abstract content tree, which corresponds to a *completely specific choice of expansion rules* for the nonterminals of the grammar. This means that the abstract tree completely determines a set of associated ground KB relations. For instance AT1 determines the set {df(diprox,tablet), da(diprox,swallow)}. These are the facts that the author asserts to be true in the actual world.

Light semantics. Such facts are aspects of the document content that the document "exports" to the knowledge base and thereby makes formally explicit. They provide what we shall call a *light semantics* for the document. In terms of light semantics, if we were to build a standard logical form for the whole document, for instance for AT1, that logical form would simply be the conjunction of the associated asserted facts df(diprox,tablet) $\wedge$da(diprox,swallow). Light semantics does not attempt to model the whole semantics of the document (for instance, in our example, there is no explicit logical counterpart to the different choices for the comment nonterminal), but focuses instead on modeling those parts of the document semantics that can be tractably handled both by the knowledge representation component and by the authoring process.

Knowledge acquisition. Once the author has committed to a document, he has revealed a certain number of facts that he knows about the actual world and that the KB possibly did not "know". These facts (in our example: df(diprox,tablet) and da(diprox,swallow)) can then be added to the ABOX of the knowledge base, and can be used either for their own sake (knowledge acquisition) or in order to constrain the authoring of a new document.

So after the authoring of AT1, the ABOX of KB1 becomes:

ABOX :
df(burpal,solution)
da(burpal,drink)
df(diprox,tablet)
da(diprox,swallow)

Suppose now the user authors a new document, first making a selection for drug(D), and choosing diprox. Then the KB "knows" that tablet is the only choice for F and swallow the only choice for A. Indeed they are possible choices (because df(diprox,tablet) and da(diprox,swallow) are in the ABOX of the KB), but are also the *only* choices, for diprox is now known to be in TabletDrugs and in SwallowDrugs; it can therefore not be in SolutionDrugs or in ChewDrugs or in DrinkDrugs, which means that none of the facts df(dirprox,solution), da(diprox,chew) or da(diprox,drink) may hold. After the author's choice of diprox, the derivation frontier is form(F), admin(A), comments(diprox,A) with the constraint df(diprox,F). The author then chooses to expand form(F), and the system notices that the only live choice is tablet, and performs this expansion without asking the user. The frontier is now admin(A), comments(diprox,A), with the constraint df(diprox,tablet). Now the user can choose to expand admin(A), and the only live choice is swallow. At that point the frontier is comments(diprox,swallow) with the constraint df(diprox,tablet). The author can then make choices for comments(diprox,swallow) that lead to zero or several instances of comment(diprox,swallow). At a certain point he will choose the nonrecursive expansion com1, which will lead to an empty frontier, with the constraints df(diprox,tablet) and da(diprox,swallow).

We could obviously suppose here that rather than waiting for the user to point to the nonterminal he wants to expand next before finding the live choices for this nonterminal, the system could find all the live choices for all nonterminals on the frontier beforehand, and do the obligatory expansions without any input from the user, but at a slightly higher computational cost. In this way, after the initial choice of diprox as the drug, the other steps of the authoring process would be done automatically, apart from the choice of how many (and which) comments to make, which would still remain the responsibility of the author.

4 Conclusion

In the course of the paper we have defined different notions such as *live-death* issues in authoring processes, *closed-world* versus *open-world* authoring, and *light document semantics*. We have presented a formal approach to closed-world authoring that shows a correspondence between life-death problems and conjunctive Datalog queries, as well as a formal approach to open-world document authoring that stresses the strong connections with current work on hybrid knowlege bases of the Datalog-DL type. We have also sketched proofs of decidability for life/death issues in these different processes. Finally we have shown how an open-world authoring context can be used for supporting a novel form of knowledge acquisition.

Acknowledgments

Thanks to Jean-Marc Andréoli, Caroline Brun, Pierre Isabelle, Aaron Kaplan, Aurélien Max and Sylvain Pogodalla for inspiration and discussions and to the anonymous KRDB-02 reviewers for suggestions about improving the presentation.

References

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] C. Brun, M. Dymetman, and V. Lux. Document structure and multilingual authoring. In *Proc. of First International Natural Language Generation Conference (INLG)*, 2000.

[3] S. Ceri, G. Gottlob, and L. Tanca. *Logic Programming and Databases*. Springer-Verlag, 1989.

[4] José Coch and Karine Chevreau. Interactive multilingual generation. In A. Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing, LNCS 2004*. Springer, 2001.

[5] Giuseppe De Giacomo. *Decidability of Class-Based Knowledge Representation Formalisms*. PhD thesis, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", 1995.

[6] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. Reasoning in description logics. In Gerhard Brewka, editor, *Principles of Knowledge Representation*, pages 191–236. CSLI Publications, Stanford, California, 1996.

[7] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. AL-log: Integrating datalog and description logics. *Journal of Intelligent Information Systems*, 10(3):227–252, 1998.

[8] Marc Dymetman, Veronika Lux, and Aarne Ranta. XML and multilingual document authoring: Convergent trends. In *Proc. Computational Linguistics COLING*, 2000.

[9] Malte Gabsdil, Alexander Koller, and Kristina Striegnitz. Building a text adventure on description logic. In *Proceedings of KI-2001 Workshop on Applications of Description Logics*, Vienna, 2001.

[10] Erich Grädel, Martin Otto, and Eric Rosen. Two-variable logic with counting is decidable. In *Proc. of the 12th IEEE Symp. on Logic in Computer Science (LICS'97)*, pages 306–317. IEEE Computer Society Press, 1997.

[11] Alon Y. Levy and Marie-Christine Rousset. CARIN: A representation language combining horn rules and description logics. In *European Conference on Artificial Intelligence*, pages 323–327, 1996.

[12] Cécile Paris, Keith Vander Linden, Markus Fischer, Anthony Hartley, Lyn Pemberton, Richard Power, and Donia Scott. A Support Tool for Writing Multilingual Instructions. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI) 1995*, pages 1398–1404, Montréal, Canada, 1995.

[13] F. Pereira and D. Warren. Definite clauses for language analysis. *Artificial Intelligence, 13:231 – 278, 1980.*, 1980.

[14] Richard Power and Donia Scott. Multilingual authoring using feedback texts. In *COLING-ACL*, pages 1053–1059, 1998.

[15] Aarne Ranta. Grammatical framework work page, 1999—. www.cs.chalmers.se/~aarne/GF/pub/work-index/index.html.

[16] Aarne Ranta. Grammatical Framework: a Type-Theoretical Grammar Formalism, 2002. http://www.cs.chalmers.se/~aarne/work/gf-jfp.ps.gz.

[17] K. Striegnitz. Model checking for contextual reasoning in nlg, 2001. ICOS-3. Inference in Computational Semantics Workshop. Siena.