THE TREATMENT OF NEGATION IN LOGIC PROGRAMS FOR REPRESENTING LEGISLATION

Robert Kowalski
Imperial College, London

Abstract.

Logic programs represent knowledge in the form of *implications*

A if B_1 and ... B_n, $n \geq 0$

where the *conclusion* A is an atomic formula and each *condition* B_i is either an atomic formula or the negation of an atomic formula. Any variables are assumed to be universally quantified, with a scope which is the entire sentence. A negated condition "not A_i" is deemed to hold if the corresponding positive condition A_i can be shown to fail to hold. This interpretation of negative conditions is called negation by failure (NBF) [Cl 78]. It has the characteristic that only the positive "if-half" of a definition needs to be given explicitly. The negative "only-if" half is given implicitly by NBF.

The obvious problem with NBF is that it supplies the only-if halves of implications, whether or not they are intended. I shall discuss a possible solution to this problem in the context of discussing the more general problem of representing negative conclusions. I shall focus on examples taken from our formalisation of the 1981 British Nationality Act (BNA) [SSKKHC 86]. I shall argue that many negative sentences can be regarded as integrity constraints and consequently can be eliminated by transformations such as those developed by Asirelli et al [ASM 85] and Kowalski and Sadri [KS 88]. Among such sentences are ones expressing prohibitions. The interpretation of prohibitions as integrity constraints suggests a possible approach to the treatment of deontic modalities.

Example: Deprivation of citizenship.

Part V, section (40) of the BNA concerns deprivation of citizenship. Subsections (1) and (3) specify two situations where the Secretary of State may deprive a person of British citizenship. Both start out in the same way:

> "Subject to the provisions of this section, the Secretary of State may by order deprive any British citizen to whom this subsection applies of his British citizenship if the Secretary of State is satisfied that ..."

This establishes the logical form of subsections (1) and (3) as implications having positive conclusions. Subsections (2) and (4) specify to whom subsections (1) and (3) apply, and therefore have the effect of defining one of the conditions of the implications of subsections (1) and (3).

Subsection (5), however, is a negative statement:

> "The Secretary of State -
> (a) shall not deprive a person of British citizenship under this section unless he is satisfied that it is not conducive to the public good that that person should continue to be a British citizen;"

Whereas the meaning of (1) and (3) have the form

A if B

the meaning of (5) has the form of a denial

not (A and not C),

where "unless C" is understood as "if not C" and where "the Secretary of State shall not deprive ..." is understood as the negation of "the Secretary of State may deprive ...".

Following [ASM 85], we have shown [KS 88] that the denial can be eliminated, by adding extra conditions to the implications (1) and (3), obtaining new implications of the form:

A if B and C.

Intuitively, the extra condition, added to every implication having A as its conclusion, guarantees that A will not hold unless C also holds.

The advantage of the transformation is that the resulting implications have the form of a logic program which can be executed in a simple manner. The advantage of the original formulation is that it is a more natural statement of the problem domain - closer to a program specification than to a program.

It might be tempting to represent (5) directly as part of the formalisation of the legislation, on the same level as the formalisation of (1) and (3). The problem with this is that there might then be circumstances under which both A and not A would be derivable. Such a formalisation would fail to capture the intention of the legislation that, where such a contradiction might arise, the conclusion A should be withheld. This intention is signaled by the phrase "subject to the provisions of this section" in (1) and (3) and is captured by treating (5) as an integrity constraint, or equivalently by eliminating (5) by means of the transformation just illustrated.

Subsection (5) can also be interpreted as an exception to the general rules expressed in subsections (1) and (3). Our view of (5) as an integrity constraint is compatible with this view of (5) as an exception. Moreover, the distinction in [R89] between a separate representation of exceptions and a compiled representation "as a flat formalisation" is similar to our distinction between the separate representation of integrity constraints and the representation which eliminates integrity constraints by means of transformation.

Integrity Constraints.

The notion of integrity constraint arises in a database context as a sentence which must be true of all states of the database. In the context of a deductive database, which has the same syntactic form as a logic program, the notion of integrity constraint is less well established, and several proposals have been put forward, e.g. [LST86, SK88, KS88].

For our purposes, we shall regard an *integrity constraint as a sentence of first-order logic, which when expressed as denials augmented, if necessary, with auxiliary implications is consistent with the database.* This is the view taken in [SK88] and [KS88]. A *denial* is a sentence of the form

not (B_1 and ... and B_n), $n \geq 1$

where each condition B_i is either an atomic formula or the negation of an atomic formula, and any variables are assumed to be universally quantified, with a scope which is the entire sentence.

The restriction that the integrity constraint be expressible as denials together with auxiliary implications is not a limitation. Any sentence of first-order logic can be re-expressed in this form. For example, the sentence

for all x there exists y ((P(x y) or Q(x y) if R(x y))

can be reexpressed in the form

not S(x)
S(x) if R(x y) and not P(x y) and not Q(x y)

where "S" is a new predicate symbol, not occurring elsewhere in the database or the other integrity constraints. As a simpler example, an implication

C if A

can be reexpressed as a denial

not (A and not C),

which is the form of subsection (5) in our previous example.

It is instructive to compare the interpretation of subsection (5) as an integrity constraint with its interpretation as an implication of the form C if A in the database. Interpreted as an implication in the database it allows us to conclude that the Secretary of State is satisfied that it is not conducive to the public good that a person should continue to be a British citizen, whenever we are informed that the Secretary of State has deprived that person of British citizenship. This interpretation fails to constrain the behaviour of the Secretary of State in any way.

Interpreted as an integrity constraint, however, subsection (5) expresses that a violation occurs whenever the Secretary of State deprives someone of British citizenship, without the Secretary of State being appropriately satisfied concerning the public good. This violation is signalled by the derivation of an inconsistency.

Moreover, the wording of subsection (5) suggests further that, where such a violation might occur, consistency should be maintained by withholding deprivation of British citizenship rather than by changing the Secretary of State's views concerning the public good. Thus we are lead not only to interpret (5) as an integrity constraint, but to interpret it as an integrity constraint which indicates how violations of integrity are to be avoided.

The elimination of integrity constraints.

Although the theory of integrity checking is well-developed, in practice, conventional database systems perform only limited integrity checking and deductive databases (and logic programs) perform none at all. In the case of a deductive database, this is because the effect of integrity checking can often be achieved without explicit integrity constraints by transforming the original database [ASM85, KS88]. The simplest case is where an integrity constraint has been expressed in the form

not (A and C)

where both A and C are atomic formulae, and A has been nominated to be *retracted* if ever an inconsistency should arise. The transformation guarantees that the integrity constraint is maintained by withholding the conclusion A whenever the condition C holds: All implications in the database of the form

A if B,

where B is a conjunction of atomic formulae or negations of atomic formulae, are replaced by implications

A if B and not C.

The transformation can easily be generalised to deal with more general cases. In particular, if, as in the case of subsection (5), C is the negation not C' of an atomic formula C, the transformation replaces all implications of the form

A if B

by implications

A if B and C.

Example Provisions for reducing statelessness.

As the following example shows, negative statements are not restricted to statements of prohibition. Schedule 2, paragraph 1 is concerned with reducing statelessness:

> "1. - (1) Where a person born in the United Kingdom after commencement would but for this paragraph, be born stateless, then, subject to sub-paragraph (3) -
>
> (a) if at the time of the birth his father or mother is a citizen or subject of a description mentioned in sub-paragraph (2), he shall be a citizen or a subject of that description ...".

Clause (b) goes on to explain the consequence that such a person might therefore be a citizen or subject of different descriptions by virtue of different parents. Surpressing certain details, subparagraph (1) (a) has the form:

```
Status(x y) if B(x)
        and Parent(x z)
        and Status(z y)
        and Sub-para-2(y).
```

Subparagraph (2) defines the relevant types of status:

> "(2) The descriptions referred to in sub-paragraph (1) are a Dependent Territories citizen, a British Overseas citizen and a British subject under this Act."

This can be formalised by conditionless implications:

```
Sub-para-2(British-Territories-citizen)
Sub-para-2(British-Overseas-citizen)
Sub-para-2(British-subject).
```

Subparagraph (3) expresses a constraint on subparagraph (1):

> "(3) A person shall not be a British subject by virtue of this paragraph if by virtue of it he is a citizen of a description mentioned in sub-paragraph (2)."

Here the intention is to insure that, if a person can become a British-Territories citizen or British-Overseas citizen by virtue of one parent, then he does not become a British subject by virtue of the other parent. This has the form:

```
not Status(x British-subject) if
            [ Status(x British-Territories-citizen) or
              Status(x British-Overseas-citizen)].
```

This can be reformulated as two denials:

```
not ( Status(x British-subject) and
      Status(x British-Territories-citizen))

not ( Status(x British-subject) and
      Status(x British-Overseas-citizen))
```

The wording of subparagraph (3), in fact, expresses the additional information that the predicate "Status(x British-subject)" should be withheld, whenever an inconsistency would otherwise occur.

Using the transformation of [KS 88], the constraint can be eliminated by transforming the representation of subparagraph (1) into the form:

```
Status(x British-subject)
        if  not Status(x British-Territories-citizen)
        and not Status(x British-Overseas-citizen)
        and B(x)
        and Parent(x z)
        and Status(z British-subject)

Status(x y)
        if  not y = British-subject
        and B(x)
        and Parent(x z)
        and Status(z y)
        and Sub-para-2(y).
```

The transformation generates two sentences, the first of which concerns those instances of subparagraph (1) to which the integrity constraint applies, and the second of which concerns those instances for which it does not apply.

Constraints on undefined conditions.

The two, previously considered examples illustrated the elimination of constraints on predicates A completely defined in the legislation itself. The transformation used in those examples does not apply to constraints on predicates that are not completely defined, but form part of the input about particular individuals. The concept of being ordinarily resident is such a predicate. Section 50, subsection (5) describes a constraint on this predicate:

"It is hereby declared that a person is not to be treated for the purposes of any provision of this Act as ordinarily resident in the United Kingdom or in a dependent territory at a time when he is in the United Kingdom or, as the case may be, in that territory in breach of the immigration laws."

This has the form

not A if C, or equivalently
not (A and C),

where the predicate A is not defined in the Act. (However, section (7) subsection (3) does give a partial definition of "ordinarily resident"). Notice that the English wording conveys the additional information that A rather than C should be withheld whenever it would be necessary to withhold one of them to avoid an inconsistency.

Notice also that, whereas our two previous examples could be considered as examples of rules with exceptions, this is an example of an exception without any rules.

Theoretically, the transformation of [KS88] applies, not only to rules already present in the legislation, but also to input of the form

A

about particular individuals, transforming the input into the form

A if not C.

But this is equivalent to explicitly checking the integrity of A without having an explicit representation of the integrity constraint.

An alternative is to use the transformation developed by Minker and his colleagues, e.g. [CGM 88], to replace all implications of the form

B if A and D

having A as a condition, by implications of the form

B if A and not C and D.

This has the effect of ensuring that, although the input of A might violate integrity, no such violation is allowed to propagate elsewhere. Indeed, the transformation would also apply to all queries

A and D?

having A as a condition, transforming them into the form

A and not C and D?

Thus, even if the input of A were to violate any integrity constraints, it would not contribute to the derivation of any consequences.

For the concept of being ordinarily resident this transformation applies, for example, to section (50), subsection (2), which defines the concept of being settled in the U.K. or a dependent territory, as well as to section (7), subsection (2), which defines entitlement to register as a citizen of the U.K. by virtue of residence.

One of the difficulties with this second transformation is that, in a large and complex text, it is generally harder to find all places where a predicate A occurs as a condition than it is to find all places where it occurs as a conclusion; and consequently it is harder to find all implications that need to be replaced by the transformation. However, as the present example shows, it can be applied in situations where the first transformation cannot.

Constraints on predicates not occurring in the legislation.

A more extreme situation arises when a constraint concerns predicates that occur neither as conclusions nor as conditions in the legislation. Two such examples in the BNA concern the interaction between the legislation and the environment in which the legislation is implemented:

"44-(2) The Secretary of State, a Governor or a Lieutenant-Governor, as the case may be, shall not be required to assign any reason for the grant or refusal of any application under this Act the decision on which is at his discretion;"

"44-(3) Nothing in this section affects the jurisdiction of any court to entertain proceedings of any description concerning the rights of any person under any provision of this Act".

It is impossible to capture the meaning of such constraints in the form of positive implications. Such examples provide further motivation for extending logic programs by the inclusion of explicitly stated integrity constraints. Explicit representation of integrity constraints is a feature of deductive databases, (as discussed for example in [LST 86] and SK 88]) and has also been proposed as an extension of logic programming [EK 89].

Negation for explicit representation of only-if halves of definitions.

In addition to its occurrence in the kinds of examples already considered, negation is also used in the text of legislation to explicitly express the only-if halves of definitions. Many of these explicit occurrences of negation simply take the form of a phrase

"otherwise not"

following the if half of a definition. Other occurrences of negation are more elaborate statements located physically apart from their if halves. A good example of this is section (37), subsection (4) which expresses that the Act provides an exhaustive definition of the different forms of Commonwealth citizenship (British citizenship, British Dependent Territories citizenship, British Overseas citizenship) and the status of British subject:

"(4) After commencement no person shall have the status of a Commonwealth citizen or the status of a British subject otherwise than under this Act".

Because there are so many different clauses defining the different forms of Commonwealth citizenship and the status of British subject, it would not be practical to express their definitions in if-and-only-if form.

The example of (37) (4) suggests how NBF could be modified to overcome the problem that it supplies the only-if half of a definition whether or not it is intened: Simply require that, whenever all clauses of the if-half of the definition of a predicate have been given, then an explicit declaration be made that there are no others. NBF can then be restricted to the demonstration of negative conditions whose predicates have been so declared.

Conclusions.

I have considered several ways in which negative statements can arise in legislation. Although many of these can be regarded as exceptions to general rules, others seem to be exception without general rules. Both types of exception, however, can be regarded as integrity constraints, and in many cases can be eliminated by transformations which represent the legislation in the form of a logic program. Other uses of negation, which can not be eliminated, motivate extending logic programs to include explicit statements of integrity constraints or explicit declarations that the if-halves of definitions have been completed.

Many of the negative statements occurring in legislation express prohibitions. It seems that they can be regarded as integrity constraints, whether or not they can be eliminated by transformations. It is interesting to consider whether positive statements expressing obligations might also be regarded as integrity constraints. These possibilities are interesting topics for further research.

Acknowledgement.

This research has been supported by the Science and Engineering Research Council as part of the Alvey Programme. I am grateful to Fariba Sadri for helpful discussions about this work.

References.

[ASM 85] Asirelli, P., De Santis, M. and Martelli, M. [1985]: "Integrity Constraints in Logic Databases", J. Logic Programming, Vol. 2, No. 3, pp. 221-232.

[Cl 78] Clark, K. L. [1978]: "Negation as failure", in "Logic and Databases", Gallaire, H. and Minker, J. [Eds], Plenum Press, pp. 293-322.

[CGM 88] Chakravarthy, U.S., Grant, J. and Minker, J. [1988]: "Foundations of Semantic Query Optimization for Declarative Databases". In "Foundations of Deductive Databases and Logic Programming", J. Minker [Ed.] Morgan Kaufmann Publishers, Los Altos, Ca.

[EK 89] Eshghi, K. and Kowalski, R. A. [1989]: "Abduction Compared with Negation by Failure", Proceedings of the Sixth International Logic Programming Conference. MIT Press.

[KS 88] Kowalski, R.A. and Sadri, F., [1988]: "Knowledge Representation without Integrity Constraints", Department of Computing, Imperial College, University of London.

[LST 86] Lloyd, J.W., Sonenberg, E.A. and Topor, R.W. [1986]: "Integrity Constraint Checking in Stratified Databases", "J. Logic Programming, Vol. 4, No. 4, pp. 331-343.

[R89] Routen, T., [1989]: "Hierarchically Organised Formalisations". In Proceedings of the Second International Conference on Artificial Intelligence and Law. ACM publications.

[SK 88] Sadri, F. and Kowalski, R.A., [1988]: "A Theorem-Proving Approach to Database Integrity". In "Foundations of Deductive Databases and Logic Programming, J. Minker [Ed.], Morgan Kaufmann Publishers, Los Altos, Ca.

[SSKKHC 86] Sergot, M. J., Sadri, F., Kowalski, R. A., Kriwaczek, F., Hammond, P. and Cory, H. T. [1986]: "The British Nationality Act as a Logic Program", CACM, Vol. 29, No. 5, pp. 370-386.