PROCEEDINGS OF THE INTERNATIONAL CONFERENCE
ON FIFTH GENERATION COMPUTER SYSTEMS 1992,
edited by ICOT. © ICOT, 1992

A Semiformal Metatheory for Fragmentary and Multilayered Knowledge as an Interactive Metalogic Program

Andreas Hamfelt and Åke Hansson

Uppsala Programming Methodology and Artificial Intelligence Laboratory
Computing Science Dept., Uppsala University
Box 520, S-751 20 Uppsala, Sweden
+46-18-18 25 00
hamfelt@csd.uu.se or hansson@csd.uu.se

Abstract

The expressiveness of metalogic programming enables a logically pure, structure preserving, easily updated representation of fragmentary, multilayered knowledge, which, since neither fully formalisable nor static, requires assimilation of externally supplied knowledge as well as coping with changes. In legal reasoning, e.g., only schematic rule descriptions are available from which case specific rules are dynamically specialised. Such rule proposals must be accepted by proposals for metarules of legal interpretation which, in turn, must be accepted by proposals for metametarules, etc. Inferencing between two adjacent levels corresponds to upward reflection; in contrast downward reflection is disallowed. Typically, formalising involves three distinct theories: the informal theory, the formal theory, and the *informal* metatheory discussing the latter, but for multilayered, imprecise informal theories we propose instead a *semiformal* metatheory discussing both theories. It expresses legal knowledge as a Horn clause metaprogram, implemented in Prolog, which interactively constructs and presents metaproofs (proof terms) allowing users to assess, accept or reject derived conclusions.

1. Introduction

Metaprogramming is an important technique for the three interrelated topics "knowledge representation", "knowledge processing", and "knowledge assimilation" [Kowalski 1990]. The first deals with the apt choices of formalism and approach for building formal theories, the second with the construction of proofs of theorems from such theories and/or the identification of the existence of such proofs, and the third with the assimilation of new knowledge into existing theories. We describe how these three topics and their interrelationships are involved in a semiformal metatheory characterizing fragmentary, multilayered, not fully formalisable legal knowledge as a metaprogram. This metaprogram interactively constructs metaproofs and facilitates the study of the complexity of these interrelationships as well as the adequacy of the formalisation attempt.

This study shows how *upward reflection* can be used as a powerful reasoning method. The potentials of reflection have been demonstrated in artificial domains, e.g., for representing multiagent belief and knowledge [Kowalski 1990] and for exploiting properties of predicates such as symmetry [Costantini 1990]. To our best knowledge, however, our study is the first illustrating reasoning with realistic knowledge which requires upward reflection and we hope it will contribute to the understanding of reflection as a knowledge representation tool. For instance, it indicates that, while upward reflection has its informal counterpart in legal reasoning, applying downward reflection violates on the contrary the inherent structure of legal knowledge. Upward reflection in legal reasoning is connected to "rules of legal interpretation". Briefly, if we propose a legal rule for solving a legal case we must show that the rule's structure and content are in accordance with the (meta)rules of legal interpretation, otherwise the rule is legally invalid. Likewise in automatized legal reasoning, a formula A representing a legal rule can be assumed included in an object level theory OT representing legally valid rules if its inclusion accords with the metalevel theory MT of formulas representing rules of legal interpretation, i.e., assuming *Demo* defines provability we have $Demo(OT, name(A)) \leftarrow Demo(MT, name(Demo(OT, name(A))))$ where *Demo* holds for sentences belonging to or deducible from a theory, cf. Kowalski [1990].

Though inessential in principle, metaprogramming is often convenient in practice. Reasons may be its naturalness of representing the domain knowledge or even the impracticability of giving perfect object level representations. This is paralleled in law by the role of "rules of legal interpretation" which are "inessential in principle, in the sense that, although they are necessitated in practice by the imperfections and the dynamic character of the existing systems, they would not be needed in a perfect, unambiguously formulated, consistent, and complete legal system, conformable to a stable social reality. The actual function of rules of legal interpretation is to direct the identification of the existing system and its continuous construction and readjustment." ([Horovitz 1972], p. 94). Since (meta)rules of legal interpretation are imperfect as well, metametarules are also necessary etc., giving a whole multilayered hierarchy, which may roughly be axiomatized as a multilevel theory structure in a metalogic program.

Kleene ([1980], pp. 65, 69) introduces three separate and distinct "theories" involved in the process of a formalisation: (a) the informal theory of which the formal system constitutes a formalisation, (b) the formal system or object theory, and (c) the metatheory, in which the formal system is described and studied where (b), which is formal, is not a theory in the common sense, but a system of symbols and of objects built from symbols described from (c). Theories (a) and (c), which are informal, do not have an exactly determined structure, as does (b). Consider the following two approaches for studying (b): (i) the formal theory (b) is "introduced at once in its full-fledged complexity" and investigated by methods without making use of an interpretation. (This is known as the metamathematical approach if the methods are finitary.) (ii) the formal theory (b) is studied by recognizing an interpretation of the theory under which it constitutes a formalisation of (a), i.e., we analyse existing informal theories (a), "selecting and stereotyping fundamental concepts, presuppositions and deductive connections, and thus eventually arrive at a formal system", i.e., at the formal theory (b).

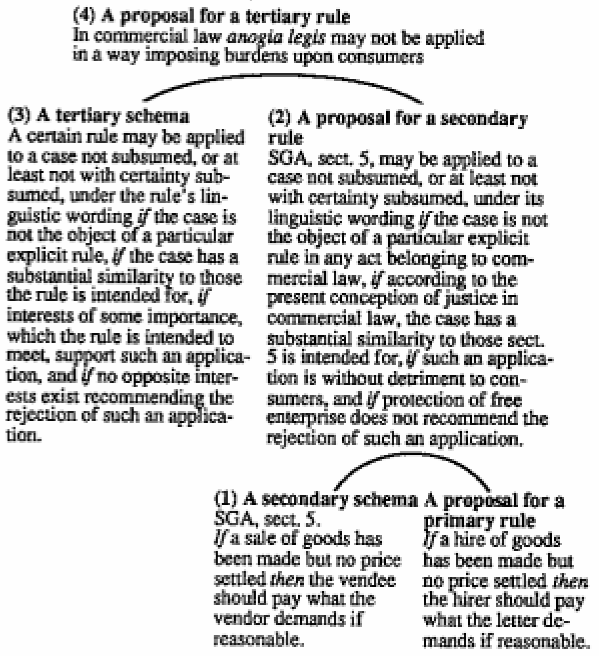
Approach (i) presupposes that the complexity of the formal theory is fully understood. This does not hold in our domain where a realistic system can only have a partial axiomatization of the formal object theory. This axiomatization can gradually be extended, though, by consulting the user both for supply of metalinguistic entities representing objects of the formal object theory and for completing formal proofs in it. Thus, in a practical system we must adhere to approach (ii).

However, an isolated study of "knowledge processing" and "representation" within the formal object theory (b) itself can be carried out along approach (i) if the problem of supplying and assimilating external knowledge is disregarded from. In such a study [Hamfelt and Hansson 1991a] we devised a Horn clause metalanguage axiomatization (c) of a formal object theory (b), as a theory of an n-level language where each level $i+1$ is the formal metalanguage of the language of level i, and where the informal theory (a) being formalised in (b) was hierarchical fragmentary legal knowledge. The investigation could be carried out as though we had had an ideal full one-to-one axiomatization (c) of the underlying formal object theory (b) by simply assuming that sufficient fragments were available for the particular cases studied, fragments that must be supplied from the outside in a "real life" application since it is impossible anticipating what knowledge will be needed.

This simplifying assumption has been removed in the present work which delves into the knowledge assimilation problem. Although the non-logical axioms of the formal object theory (b) cannot be enumerated in advance its possible content can nevertheless be discussed in a theory (c) of a metalanguage which may be informal but also formal or both. To this end we have devised a semiformal metalanguage—whose object language is the n-level language of (b)—for a theory (c) which axiomatizes the "available" part of the formal object theory (b) and encodes rules for the assimilation into it of externally supplied knowledge fragments. Knowledge assimilation is dependent on the deductive structure of (b), which can be accounted for in (c) since its objects of discourse include formal proofs, i.e., sequences of formulas of (b). Below, $\mathcal{IT}$, $\mathcal{OT}$ and $\mathcal{MT}$ denote, respectively the (a), (b) and (c) of our system.

2. The Informal Legal Theory

Let us detail our conception of the structure of legal knowledge, i.e., our informal legal theory $\mathcal{IT}$, so as to illustrate how rules at different levels operate together, as shown in fig. 2.1. We refer to [Hamfelt 1990, Hamfelt and Hansson 1991a, 1991b] for a more thorough account


(4) A proposal for a tertiary rule
In commercial law *anogia legis* may not be applied in a way imposing burdens upon consumers

(3) A tertiary schema
A certain rule may be applied to a case not subsumed, or at least not with certainty subsumed, under the rule's linguistic wording *if* the case is not the object of a particular explicit rule, *if* the case has a substantial similarity to those the rule is intended for, *if* interests of some importance, which the rule is intended to meet, support such an application, and *if* no opposite interests exist recommending the rejection of such an application.

(2) A proposal for a secondary rule
SGA, sect. 5, may be applied to a case not subsumed, or at least not with certainty subsumed, under its linguistic wording *if* the case is not the object of a particular explicit rule in any act belonging to commercial law, *if* according to the present conception of justice in commercial law, the case has a substantial similarity to those sect. 5 is intended for, *if* such an application is without detriment to consumers, and *if* protection of free enterprise does not recommend the rejection of such an application.

(1) A secondary schema
SGA, sect. 5.
If a sale of goods has been made but no price settled *then* the vendee should pay what the vendor demands if reasonable.

A proposal for a primary rule
If a hire of goods has been made but no price settled *then* the hirer should pay what the letter demands if reasonable.

Fig. 2.1. Schemata and rules.

but the below description should suffice for this paper.

Consider the provision 1 in fig. 2.1, an ordinary statutory rule from the Swedish Sale of Goods Act (sect. 5, SGA). This provision is not only applicable to sale of goods. It could, e.g., be analogically applied to, e.g., hire of goods, or extensively interpreted, or interpreted by inversion (*e contrario*), etc., and embraces thus a lot of primary rules. One but only one of these is the rule given by a literal reading of the tokens building the provision and not even this rule has a legal validity which can be taken for granted. The provision 1 is a *schema* for all these rules and since this schema is about primary rules and thus conceptually belongs to the secondary level we call it a secondary schema.

The relation between secondary schemata and primary rules is given by secondary rules. For example, the relation between the schema 1 and real primary rules is given by secondary rules such as rule 2 in fig. 2.1 which is just an example of how a secondary rule for *analogia legis* in commercial law could possibly look. In the same way there exist tertiary schemata for secondary rules and tertiary rules that give the relation between these schemata and the secondary rules, etc. The secondary rule 2 originates from the tertiary schema 3 in fig. 2.1. Information about the relation between this schema and secondary rules such as rule 2 is given by tertiary rules, such as rule 4 in fig. 2.1.

Schematic descriptions of rules at various levels are important. We have argued elsewhere [Hamfelt 1990] that a lawyer only has a schematic knowledge of legal rules; each adjudication comprises an interpretation of schemata for legal rules and results in the construction of specialised rules applicable only to the case at hand. An obvious example is our secondary rule 2 for *analogia legis* in commercial law. It is not generally applicable. The rule is the result of an interpretation at levels above the secondary and only "applicable" in an individual adjudication, i.e., in a particular legal case. In another legal case the interpretation at the levels above the secondary may yield another formulation of the rule 2. Rules, such as rule 2, are thus generated for each individual adjudication and there exists a diversity of possible formulations. What they have in common is that they all originate from a common schematic description 3 which in this case conceptually belongs to the tertiary level. The schematic description 3 originates from the legal literature. Rule 4 is thus also, in its turn a possible specialisation of a quaternary schema, proposed by quaternary rules which in their turn are specialisations of quinary schemata, etc. Fig. 2.2 illustrates the hierarchy of legal knowledge in which the levels have been made distinct by introducing, at each level i, the *names* for the rules of the level $i-1$.

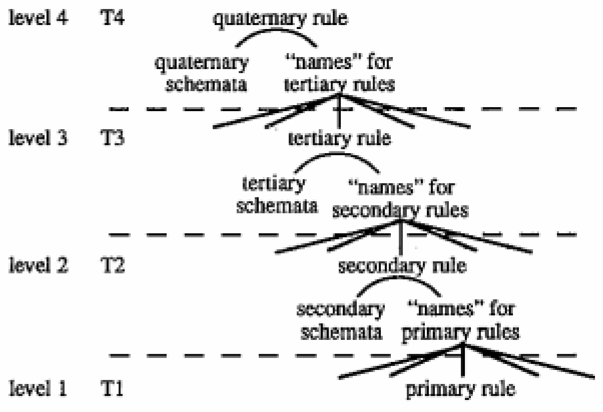


Figure 2.2. Levels of legal knowledge.

The specialisation of schemata (principles) must yield *meaningful* rules which are legally *acceptable* for the current case. The first means that the rule only consists of legal concepts embraced by the principle, the second that its content in addition is legally adequate and its premises fulfilled so it in fact applies. The latter is recursively settled, i.e., a sentence is legally acceptable if either its content directly is accepted by rules at the higher adjacent level (base case), or it follows from other acceptable sentences at its own level (recursive case). The base case is thus informal upward reflection where the upper level enforces a sentence on a lower level.

Till now rule specialisation has been described in a top-down perspective. Specialisations cannot, however, be carried out without interpretation data, the supply of which has a character of a bottom-up process, since these data mainly originate from a description of the legal case. If C is a description in legal terminology of a legal case and J is a suggestion for a judicial decision, then $J \leftarrow C$ is a proposed primary rule. Establishing that $J \leftarrow C$ is accepted by the (secondary) metarules of legal interpretation coincides then with resolving the case.

Level 1 in figure 2.2 may be understood as an object level, level 2 as its metalevel, level 3 as its metametalevel the object level of which is the pair of level 1 and level 2, etc. The structure of $\mathcal{IT}$ could be understood as a hierarchy of "theories" where the language of each level i consists of all meaningful rules expressed by schemata and conditions for their specialisation on the level $i+1$. Each T_i is characterized in T_{i+1} by schematic descriptions of its sentences and rules for deciding which specialisations of these are meaningful and legally acceptable. The same holds for the theory T_{i+1} with respect to a theory T_{i+2}, etc., and since no level has rules which do not require interpretation this proceeds *ad infinitum.* This can be handled by choosing some level n to be the "top-most" at which discretion is used for specifying schemata, thus making the validity of a primary rule proposed for resolving a case and of all rules on each level i, $1 \leq i < n$, ultimately depending on discretion.

$\mathcal{IT}$ can thus be characterized as a set of theories $\{T_1, T_2, \ldots, T_n\}$ where each theory T_i is understood as a collection of sentences fully determined by the higher adjacent theory T_{i+1}, i.e., if from theory T_{i+1} it can be deduced a theorem expressing that a sentence A_i is provable from theory T_i, then A_i belongs to theory T_i. We need a definition of the *provability relationship* between two adjacent levels $i+1$ and i ranging over n distinct levels (theories), and forming a hierarchy of dependent relations directed from the highest to the lowest level. On each level i this provability relation $T_i \vdash A_i$ should coincide with the rules of logic. The hierarchical dependence of the provability relationship, $1 \leq i < n$, may be characterized as follows $T_i \vdash A_i$ *iff* $T_{i+1} \vdash$ "$T_i \vdash A_i$", where "$T_i \vdash A_i$" names $T_i \vdash A_i$. With theory T_n specified, the hierarchy decides the content of the object level as well as of all the other levels. That is to say, T_n decides the contents of all the theories $T_1, \ldots, T_{n-1}$. The definition constitutes the *rules of acceptance* of $\mathcal{IT}$.

3. The Formal Object Theory

The provability definition in a formalisation $\mathcal{OT}$ of $\mathcal{IT}$ must support upward but prevent downward reflection. This is due to the imperfection of realistic legal systems. Both reflection principles would be allowed in Horovitz' unattainable perfect system where each T_i is an enumerable and decidable set which, for each of all the meaningful rules of its language, includes either the rule itself or its negation, but not both. Each theory T_i expresses T_{i-1} completely in a metalanguage exactly determining its content and conforms to both reflection principles since everything derivable from theory T_{i-1} can be "simulated" in theory T_i and vice versa; in principle, it would be sufficient to consider one of these theories in an ideal formalisation. In reality, however, the formalisation of each T_{i-1} can only be partial, imperfect and schematic and applying the rules in T_i is necessary for assessing, accepting or rejecting the rules in T_{i-1}. This corresponds to upward reflection, where something proved on an upper level is forced upon a lower level. Downward reflection is unsound however since something 'proved' from imperfect knowledge on a lower level cannot be forced upon an upper level thereby perhaps contradicting rules accepted by the legal principles on this level.

Thus, the amalgamated language of Bowen and Kowalski [1982] is inadequate for characterizing $\mathcal{IT}$ since its provability predicate *Demo* is founded on representability (equivalence between object language and metalanguage proofs) and requires both reflection rules.

Demo represents provability on the object level thus presupposing that an object level proof procedure exists, which is not the case in our domain whose counterpart to *Demo defines* rather than represents a provability relation for an object language. However, *Demo* predicates giving *definitional* non-conservative extensions of theories deviating from the representability notion are not unusual, e.g., metalevel definitions allowing upward reflection to enforce proofs on the object level which the object level theorem prover itself cannot carry out, cf. [Kowalski 1990]. This latter use of the *Demo* predicate accords with our understanding of $\mathcal{IT}$ and could be used to characterize its hierarchical structure.

The complexity caused by $\mathcal{IT}$'s hierarchical structure is reduced, if each T_i is characterized "locally" towards its immediate object of study, i.e., each meta/object language relation is represented separately. A link between adjacent levels is obtained by characterizing the *Demo* predicate as defining the provability relation on a lower level. For example, the formula $Demo(name(T_1), name((J \leftarrow C)))$ says (on level 2) that the primary rule $J \leftarrow C$ is provable from, and thus included in, the object theory T_1. The declarative reading of the formula is rendered by taking T_1 as a static theory implicitly consisting of all rules fulfilling the conditions for inclusion. Only the boundary between rules shown to satisfy the conditions and those not yet shown to satisfy the conditions moves just as in Sergot's "query the user" [1983] the boundary between the facts given to the computer by the user and those not yet given, moves. Stated as a goal the formula reads "is $J \leftarrow C$ provable from the theory T_1?" the proof of which corresponds to a line of arguments to the effect that the inclusion in T_1 of $J \leftarrow C$ should be regarded as in accordance with the (secondary) metarules of legal interpretation, thus showing, as hinted in sect. 2, that the rule is legally acceptable. The provability definition expresses the conditions for including $J \leftarrow C$ in T_1, i.e., the secondary rules whose inclusions in T_2 depend in their turn on theories of higher levels, whose provability definitions are characterized in a similar way yielding a whole hierarchy of interdepending provability definitions, still however allowing us to describe and consider each T_i as a separate theory.

Specifying the theory of primary rules T_1 metatheoretically in terms of what can be proved from T_1 seems thus natural. The topmost theory T_n gives an axiomatic definition of provability of theory T_{n-1} and indirectly of all theories T_i, $i < n$, hence, embracing all the non-logical axioms of these theories. We have chosen an alternative perspective where the hierarchical structure of $\mathcal{OT}$ is taken as a composite object language which can be characterized in a theory $\mathcal{MT}$ of a separate metalanguage. This metalanguage takes thus the whole n-level language of $\mathcal{OT}$ as its object language. In $\mathcal{MT}$ $Prover(Demo(name(T_1), name(A)), \ldots, \ldots, Proof)$, e.g., expresses that a formalisation of a primary rule A is included in the formal theory of $\mathcal{OT}$, which represents the informal theory T_1 of $\mathcal{IT}$. In the metalanguage this inclusion is verified by a sequence *Proof* of statements each of which names an "object/object"-inference or a "meta/object"-inference of the object language, thus constituting a metaproof in $\mathcal{MT}$ of a formal proof in $\mathcal{OT}$.

4. The Semiformal Metatheory

We now briefly describe our metatheory $\mathcal{MT}$, mainly as a program partially characterizing $\mathcal{OT}$ whose intended interpretation is $\mathcal{IT}$. Our metalogic consists of Horn clauses and the inference mechanism of Prolog. In this metalanguage $\mathcal{MT}$ is represented by formulas, i.e., in Horn clauses, and the language of $\mathcal{OT}$ by terms, i.e., terms of Horn clauses.

To the *formation rules* and *rules of inference* (including axiom schemata) of proof theory correspond the *rules of meaningfulness* and the *rules of acceptance*, respectively, both of which have a more vague character than their proof theory counterparts. It is a hard and sometimes impossible problem to state formation rules in the metatheory which can handle the vagueness of legal concepts. Therefore, it seems necessary, in contrast to proof theory, to complement with an external settlement of which rules are meaningful rules, and which rules are acceptable as true material implications. The specification of a formal theory is done from the starting-point of an informal theory which is the intended interpretation the formalism should capture. For example, a constant in the formal language of the formal theory is supposed to have a natural counterpart in the informal theory, e.g., to denote a specific individual or a specific category of individuals. In our case, we cannot predetermine this connection between a formal and an informal theory, because we cannot decide in advance all legal concepts that can be relevant in a legal system. This is something that has to be resolved from case to case. Therefore, we leave it to the user to decide the formal counterpart of his informal understanding of a legal concept, e.g., an informal legal concept "hirer" of $\mathcal{IT}$ will naturally get the constant "hirer" of the formal language of $\mathcal{OT}$. That is to say, we presuppose an immediate or autonymous relation between a symbol or symbols of the formal language and its informal counterpart. Thus, with external help a metatheory $\mathcal{MT}$ could be extended and some of the gaps in its schemata for non-logical axioms of $\mathcal{OT}$ be filled in yielding a formalisation of rules of $\mathcal{IT}$ obtained by specialising the available vague descriptions of these rules. In that way, a program representing $\mathcal{MT}$ could construct, interactively with the user, a metaproof in $\mathcal{MT}$ representing a formal proof in $\mathcal{OT}$; at least for the proposal of a legal case under consideration. In some sense this position could be understood as performing a specification in a metatheory of the formal language of a formal theory as well as deciding its non-logical axioms in parallel with a derivation of some conclusion from it. At the end of a session (of constructing metaproofs) the relevant fractions of this theory, and the formal proof of a particular legal case could be presented to the user for a final examination by extracting the formal proof out of the metaproof. How adequately these formal proofs, represented in the metatheory $\mathcal{MT}$, correspond to $\mathcal{IT}$ is a matter of external judgements. Observe, that $\mathcal{MT}$ is a *semiformal* theory in the sense that it has both a formal part, consisting of sentences represented as Horn clauses, and an informal part consisting of user interpreted sentences.

Thus, the *rules of meaningfulness* in $\mathcal{MT}$ can only partially characterize sentences of the object language, i.e., the language of $\mathcal{OT}$. In the representation in the metalanguage, we have to assume a fixed structure for designating a class of rules, i.e., a schema. Within this structure local differences must be met, i.e., different specialisations of the schema have to give different representations of sentences (rules) of the object language. These local differences are expressed by metavariables which have to be filled in by a user and satisfy certain interactively investigated typing conditions. Let us illustrate this with the program clause that characterizes a provision schema whose linguistic wording is the one specified for the metavariable Text. In the clauses below n(...) is a shorthand for name(...) for which we postulate an inverse law of naming, i.e., n(A)=n(B)→A=B. As to the problem of naming in metaprogramming, observe that all variables are metavariables; there are no object variables.

```
meaningfulsent(t(1),RuleProp1,[ModAt1,unspec],LegSet1,Text):-
  RuleProp1 = (legalcons(pay,X,Y,goods,price):-
                   and(actor1(X,goods),and(actor2(Y,goods),
                   and(unsettledprice(goods),and(demands(Y,price),
                   reasonable(price,goods)))))),
  ModAt1 = [X/vendee,Y/vendor], Types = [actor(X),actor(Y)],
  LegSet1 = [[provisionno(sga(5))|_],LegSet0],
  Text = the same text as in rule schema 1 in fig. 2.1.
  propertyping(t(1),RuleProp1,ModAt1,Types,Text).
```

In $\mathcal{IT}$ this provision is assumed open with respect to the concepts 'vendee' and 'vendor'. So, the assumed fixed structure of this provision is represented in the metalanguage as the term specified for the metavariable RuleProp1 with open places expressed by the metavariables X and Y. These variables have to be specialised interactively with the user. The predicate propertyping/5 is defined for this interaction. The metavariable ModAt1 expresses the relation between the concepts of $\mathcal{IT}$, i.e., the text of Text and its open parts, i.e., 'vendee' and 'vendor', and its formal counterpart in $\mathcal{OT}$ partly specified in RuleProp1. Thus, a proper typing carried out by the user gives a meaningful rule of the object language of level 1, represented in the metalanguage by the specialised term of RuleProp1. The metavariable LegSet1 identifies what part of level 1 in $\mathcal{IT}$ is relevant for a particular case.

The *rules of acceptance* may also only be partially characterized in the metalanguage. However, a user can interactively add interpretation data, thereby extending the partial characterization of $\mathcal{OT}$ in the theory $\mathcal{MT}$ of the metalanguage. What is hard characterizing is the determination of whether or not a meaningful rule belongs to a theory of $\mathcal{IT}$, i.e., is legally acceptable, and thus should have a formal counterpart in $\mathcal{OT}$. Presently, this is solved by assuming in $\mathcal{MT}$ that a rule is acceptable when a user tries to apply it, and the conditions for its application are accepted, i.e., either follow by logic from other accepted rules or are included in the theory by rules at the higher adjacent level in co-operation with the user. So, we presuppose that it is only the user who can determine the relevance of a specific principle. Consequently, at the end of a session these assumptions should be possible for a user to examine.

These aspects are encoded in the prover clause [UP] (short for upward reflection). Observe that the prover clauses belong to $\mathcal{MT}$ which takes as object theory the whole multilayered $\mathcal{OT}$. Their first demo argument defines the formalisation in $\mathcal{OT}$ of logic provability between a theory T_i of $\mathcal{IT}$ and a sentence of $\mathcal{IT}$ but though e.g., the fourth proof term argument has a counterpart in $\mathcal{OT}$— a formal proof extending over the whole hierarchy of $\mathcal{OT}$—it includes expressions solely of $\mathcal{MT}$ as well.

```
[UP]
prover(demo(n(t(I)),n(SentPropI)),ModI,LegSetI,ProofI):-
  proposesent(t(I),SentPropI,ModI,LegSetI),
  J is I + 1,
  ground([SentPropI,ModI,LegSetI]),
  permissible(t(I),SentPropI),
  prover(demo(n(t(J)),n(demo(n(t(I)),n(SentPropI)))),
          [ModAtJ,ModI],[LegSetAtJ,LegSetI],ProofJ),
  ProofI = (sentenceof(theory(I),SentPropI):-
               proofof(theory(J),proved(theory(I),SentPropI),ProofJ)).

permissible(t(I),SentPropI):-I = 1.
permissible(t(I),SentPropI):-I ≥ 2,\+ SentPropI = (Head:-Body).
```

Clause [UP] encodes in $\mathcal{MT}$ upward reflection between two theories T_i and T_j of arbitrary adjacent levels in $\mathcal{IT}$, with formal counterparts t(I) and t(J) in $\mathcal{OT}$. A sentence is assumed to belong to a theory T_i if this accords with the rules of theory T_j of the higher adjacent level. In $\mathcal{MT}$, LegSetI and ModI identify and modify formula schemata corresponding to known fragments of sentences of the theory T_i. The predicate proposesent/4 is defined to specialise interactively with a user such meaningfulsent schemata. ProofI is a metaproof in $\mathcal{MT}$ of the existence of a sequence of formulas in $\mathcal{OT}$'s formalisation of $\mathcal{IT}$ constituting a formal proof of the proposed sentence.

Upward reflection must be constrained. If each sentence were upward reflected directly when proposed, the reasoning process would ascend directly to the topmost level since the metarule proposed for assessing the sentence would itself directly be upward reflected, etc. Therefore, at levels i, $i \geq 2$, only sentence proposals which are ground facts may be upward reflected, postponing the assessment of rules, which may only be proposed as non-ground conditional sentences, till facts are activated by their premises. Under this reasoning scheme the content of all sentences involved in the reasoning process will eventually be assessed. The restriction is maintained by the permissible subgoal.

Clause [ANDI] handles ∧-introduction. In $\mathcal{MT}$ a theory T_i of $\mathcal{IT}$, with t(I) as formal counterpart in $\mathcal{OT}$, is assumed to include a sentence which is a conjunction if both its conjuncts may be assumed included in T_i.

```
[ANDI]
prover(demo(n(t(I)),n(and(G1,G2))),
        [[ModG1,ModG2],ModsBelow],LegSetI,ProofI):-
  I ≥ 2,
  prover(demo(n(t(I)),n(G1)),[ModG1,ModsBelow],LegSetI,ProofG1),
  prover(demo(n(t(I)),n(G2)),[ModG2,ModsBelow],LegSetI,ProofG2),
  ProofI =(sentenceof(theory(I),and(G1,G2)):-
              and(proofof(theory(I),G1,ProofG1),
              proofof(theory(I),G2,ProofG2))).
```

Clause [MP] encodes our version of modus ponens. In $\mathcal{MT}$ a theory T_i of $\mathcal{IT}$, with t(I) as formal counterpart in $\mathcal{OT}$, is assumed to include a sentence which is the consequence of a proposed implication of T_i whose antecedent can be assumed included in T_i.

```
[MP]
prover(demo(n(t(I)),n(HeadI)),ModI,LegSetI,ProofI):-
  I ≥ 2,
  proposesent(t(I),(HeadI:-BodyI),ModI,LegSetI),
  prover(demo(n(t(I)),n(BodyI)),ModI,LegSetI,ProofBodyI),
  ProofI = (sentenceof(theory(I),HeadI):-
              and(ruleof(theory(I),(HeadI:-BodyI)),
              proofof(theory(I),BodyI,ProofBodyI))).
```

The knowledge of rules in $\mathcal{IT}$ for assessing sentence proposals for the adjacent lower level theory T_i will at some level j be too rudimentary for composing a theory T_j. At this level, T_j is considered to be the user's opinion of the sentences proposed for T_i. This is encoded in $\mathcal{MT}$ in the clause [TOP].

```
[TOP]
prover(demo(n(t(J)),n(demo(n(t(I)),n(RulePropI)))),
       ModJ,LegSetJ,ProofJ):-
  J ≥ 2,
  \+ proposesent(t(J),(demo(n(t(I)),n(RulePropI)):-BodyJ),
                 ModJ,LegSetJ),
  externalconfirmation(t(I)),RulePropI,ModJ,LegSetJ),
  ProofJ = externallyconfirmed(sentenceof(theory(I),RulePropI)).
```

Let us now partially trace the computation of a sample query

```
>prover(demo(n(t(1)),n(RuleProp1)),Mod1,LegSet1,Proof1).
```

This query could be read as "is there a metaproof Proof1 stating that the theory T_1 of level 1 includes a primary rule which is represented in $\mathcal{OT}$ by RuleProp1 and modified by Mod1 in the legal setting LegSet1?" Since it is completely unspecified at this point what particular problem to solve the query can be stated in these general terms and be generated by the system. The goal resolves with the prover clause [UP] leading to six subgoals, the last of which builds the proof term to bind Proof1. Below, we refrain from discussing how the proof term is built during the computation. The first subgoal of [UP] is

```
proposesent(t(1),SentProp1,Mod1,LegSet1)
```

which through user interaction selects a legal rule and modifies it for the current case. The unifying clause

```
proposesent(Theory,RuleProp,Mod,LegSet):-
  (Theory = t(1);RuleProp=(demo( _,_):-Body)),[1]
  findlegalsetting(Theory,LegSet),
  meaningfulsent(Theory,RuleProp,Mod,LegSet,Text).
```

identifies the relevant part of the legal domain from which it retrieves a proposal for a rule provided it is meaningful. The latter is sorted out by meaningfulsent clauses, say, the one presented above. In this clause the propertyping condition is intended to promote that user proposed modifications preserve the rule's meaningfulness. Suppose now that the user interaction makes the first subgoal of [UP] return with the following ground argument bindings, i.e., the schemata from sect. 5 Sale of Goods Act is adapted into a primary rule proposal regulating a case of 'hire of goods',

```
LegSet1 =
  [[provisionno(sga(5)),
    provisioncategory('Determination of Purchase Money'),
    legalfield('Commercial Law')],unspec], call it ⟨legset1⟩
```

[1] The legal setting may be assumed unknown if any of these two conditions hold.

```
Mod1 = [[hirer/vendee,letter/vendor],unspec], call it ⟨mod1⟩
RuleProp1 =
  (legalcons(pay,hirer,letter,goods,price):-
     and(actor1(hirer,goods),and(actor2(letter,goods),
     and(unsettledprice(goods),and(demands(letter,price),
     reasonable(price,goods)))))), call it ⟨ruleprop1⟩
```

Now it must be established whether it accords with the higher adjacent level, i.e., the theory T_2, to assume a primary rule with this proposed content is included in the theory T_1. This is accomplished through "upward reflection". Before a formula with content information is upward reflected it must be checked for groundness (a hack) and permissibleness. These are the tasks of the third subgoal of [UP] (where ⟨*name*⟩ is shorthand for an occurrence of the term named by *name*).

```
ground([⟨ruleprop1⟩,⟨mod1⟩,⟨legset1⟩])
```

and of the fourth subgoal of [UP]

```
permissible(t(1),⟨ruleprop1⟩),
```

which permits a conditional rule on level 1 to be upward reflected. The fifth, "upward reflection", subgoal of [UP]

```
prover(demo(n(t(2)),n(demo(n(t(1)),n(⟨ruleprop1⟩)))),
       [ModAt2,⟨mod1⟩],[LegSetAt2,⟨legset1⟩],Proof2),
```

resolves with the prover clause [MP] leading to four subgoals (the first and last of which controls the index of the current level and builds the proof term, respectively). Now a secondary rule must be proposed for assessing the lower level expression. The second subgoal of [MP] is

```
proposesent(t(2),(demo(n(t(1)),n(⟨ruleprop1⟩)):-Body2),
            ⟨mod2⟩,⟨legset2⟩),
```

where ⟨mod2⟩ is [ModAt2,⟨mod1⟩], ⟨mod1⟩ is [⟨modat1⟩,unspec], ⟨modat1⟩ is [hirer/vendee,letter/vendor] and ⟨legset2⟩ is [LegSetAt2, ⟨legset1⟩].

Suppose the user chooses the *analogia legis* principle. The relation between primary rules of theory T_1 and secondary rules for analogia legis of theory T_2 is encoded in this clause:

```
meaningfulsent(t(2),RuleProp2,Mod2,LegSet2,Text):-
  RuleProp2 =
    (demo(n(t(1)),n(RuleProp1)):-
       analogialegis(n(RuleProp1),n(ModAt1),LegSet1)),
  Mod2 = [_,[ModAt1,_]],
  LegSet2 = [[interpretationtheory('analogia legis')|_],LegSet1],
  Text = '"A primary rule proposal is legally valid (i.e., belongs to the
          theory t1 of valid primary rules) if its inclusion accords with
          the secondary rule for analogia legis."...',
  propertyping(t(2),RuleProp2,[],[],Text).
```

The second subgoal of [MP] returns with its second argument bound to

```
(demo(n(t(1)),n(⟨ruleprop1⟩)):-
   analogialegis(n(⟨ruleprop1⟩),n(⟨modat1⟩),⟨legset1⟩))
```

and LegSetAt2 bound to [interpretationtheory('analogia legis')|_]. The third subgoal of [MP] is

```
prover(demo(n(t(2)),n(analogialegis(n(⟨ruleprop1⟩),
                                    n(⟨modat1⟩),
                                    ⟨legset1⟩))),
       ⟨mod2⟩,⟨legset2⟩,ProofBody2),
```

which recursively calls [MP]. Now a meaningful proposal for an actual analogia legis secondary rule will, by the second proposesent subgoal of [MP], be retrieved from this clause

```
meaningfulsent(t(2),RuleProp2, _,LegSet2,Text):-
  RuleProp2 =
    (analogialegis(n((Cons:-Ante)),n(ModAt1),LegSet1):-
      and(not(casuisticalinterpretation(LegalField,
                                         n((not(Cons):-Ante)))),
      and(intendedfor(ProvisionNo,n(TypeCase)),
      and(substantialsimilarity(n(TypeCase),n(Ante),n(ModAt1)),
      and(intendedtomeet(ProvisionNo,Interests,LegalField),
      and(supports(ProvisionNo,n(ModAt1),ProInt,Interests),
      and(recommendrejection(ProvisionNo,n(ModAt1),
                             ContraInt,Interests),
      outweigh(ProInt,ContraInt))))))),
  LegSet2 = [[interpretationtheory('analogia legis')|_],LegSet1],
  LegSet1 = [[provisionno(ProvisionNo), _,legalfield(LegalField)], _],
  Text = the same text as in rule schema 3 in fig. 2.1.
  propertyping(t(2),RuleProp2,[],[],Text).
```

with these bindings (where ⟨ruleprop1⟩ is (⟨consrule1⟩:-⟨anterule1⟩))

```
analogialegis(n((⟨consrule1⟩:-⟨anterule1⟩)),n(⟨modat1⟩),⟨legset1⟩):-
  and(not(casuisticalinterpretation('Commercial Law',
                                     n((not(⟨consrule1⟩):-⟨anterule1⟩)))),
  and(intendedfor(sga(5),n(TypeCase)),
  and(substantialsimilarity(n(TypeCase),n(⟨anterule1⟩),n(⟨modat1⟩)),
  and(intendedtomeet(sga(5),Interests,'Commercial Law'),
  and(supports(sga(5),n(⟨modat1⟩),ProInt,Interests),
  and(recommendrejection(sga(5),n(⟨modat1⟩),ContraInt,Interests),
  outweigh(ProInt,ContraInt))))))).
```

Now it must be proved that with the proposed content the antecedent of the *analogia legis* rule (call it ⟨albody⟩) is included in T_2. The third subgoal of [MP] is

```
prover(demo(n(t(2)),n(⟨albody⟩)), _,
        [[interpretationtheory('analogia legis')],⟨legset1⟩], _),
```

and each of the conjuncts in ⟨albody⟩ will be demonstrated in turn by the prover clauses [ANDI], [MP], and [UP]. To illustrate how user proposed content for a sentence is accepted (or rejected) at higher levels let us focus on the fourth conjunct which gives rise to the goal

```
prover(demo(n(t(2)),
                n(intendedtomeet(sga(5),Interests,'Commercial Law'))),
        _,[[interpretationtheory('analogia legis')|_],⟨legset1⟩]).
```

An "intended to meet" sentence must be proposed by the user. The result may be a meaningful fact (unconditional sentence) whose inclusion in the theory T_2 must be accepted by the rules of theory T_3 or it may be a rule (conditional sentence) which is assumed included in T_2 directly after the user's acceptance. The resolving clauses in the respective cases are [UP] and [MP]. Thus, in the first case upward reflection occurs immediately. In the second case upward reflection is postponed until backward inferencing by modus ponens at the current level leads to the proposal of a fact. Note that this guarantees that the application of the originally proposed rule is not accepted unless all the components of its antecedent eventually are assessed and accepted.

Suppose a fact is proposed. The goal will resolve with the prover clause [UP], whose recursive fifth subgoal resolves with the prover clause [MP] leading to the application of tertiary rules for assessing the proposed (secondary) fact. Reasons of space force us to remove a part of the trace here. The inferencing at the tertiary level is similar to that just described for the secondary level. We conclude this section with a fragment of the trace in which a tertiary fact is proposed but no quaternary rules exist for assessing it. The upward reflected goal looks like

```
prover(demo(n(t(4)),
            n(demo(n(t(3)),
                   n(adequatetoequalize(
                        'actors with similar economical positions',
                        'consumer protection'/'hirer protection',
                        'Commercial Law')))),
        Mod4,LegSet4, _).
```

For the theory T_4 proposesent fails however to return any quaternary rules which may assess the adequatetoequalize fact. The goal resolves with the prover clause [TOP] and the user may or may not accept the content of the "adequate to equalize" rule.

Provided the rule is accepted this completes the computation of the fourth conjunct in the antecedent of the analogia legis rule. The following three conjuncts in the antecedent of the analogia legis rule are computed likewise which completes the computation of the initial query. A conclusion is not considered as final before the line of arguments leading up to it has been considered and accepted by the user. To this end the user needs a comprehensible presentation of the proof term. We illustrate elsewhere [Hamfelt and Hansson 1991b] how derivations of goals can be entrusted to the user's acceptance or rejection by an interactive piecemeal unfolding of a term representing the proof of the goal.

5. Coping with Change

A program should be able to *cope with changes* in the frequently revised legal knowledge it formalises. Also it should be *structure preserving* ("isomorphic") modulo this knowledge, cf. [Sergot *et al.* 1986]. This is a conflict, Bratley *et al.* [1991] claim, since coping with changes requires modifying "implicit or explicit rules which do not correspond directly to paragraphs in the text of law".

Our metalogic program $\mathcal{MT}$, however, is a structure preserving formalisation of legal knowledge coping with changes. The schemata give a modular, direct and easily changed description of statutory rules and (meta ...)metarules of legal interpretation. $\mathcal{MT}$ is modular both horizontally and vertically entailing that adjustments can be made locally to the schemata for the (higher level) rules of legal interpretation as well as to the schemata for the ordinary (low level) statutory rules. The level of the knowledge is identified and the appropriate adjustment made to its rule schemata, which then control the computation of accepted rules assumed included in theories of the lower adjacent level. Also, since $\mathcal{MT}$ takes as its object language the whole n-level language of $\mathcal{OT}$, we can encode in the formal part of $\mathcal{MT}$, rules coping with global changes which are not possible to localize to rule schemata of a certain level. Furthermore, if the legal system has undergone an even more drastic revision, a large part of our system will nevertheless remain intact since the structure of principles such as *analogia legis* will hardly be affected. The structure preserving model of the British Nationality Act [Sergot *et al.* 1986] is according to Kowalski and Sergot [1990] "of limited practical value" since it expresses a "layman's reading of the provision" but in our $\mathcal{MT}$ expert knowledge may be incorporated e.g., for verifying the correctness of $\mathcal{OT}$, modifying and augmenting it, and for suggesting promising ways for applying its rules.

6. Related Work

Allen and Saxon [1991] discuss, in contrast to our multiple semantic interpretations, assistance for multiple structural interpretation of components of provisions, such as "if", "not", "provided that", e.g., by changing which component is taken as the main connective of a sentence. The logical relationship between theories comprising interpretative knowledge and interpreted theories is not analysed.

Assessing and compiling persuasive lines of arguments pro and contra different, often contradictory, legal decisions is important in legal reasoning. Proof terms should thus be objects of discourse and be reasoned about, which they are in $\mathcal{MT}$. This is advocated also by Bench-Capon and Sergot [1988], who do not, however, propose a formalisation or a detailed informal theory, such as our $\mathcal{IT}$, concerning how these aspects are sorted out in informal legal reasoning.

7. Conclusions and Further Work

Above we have proposed a *novel approach* for representing fragmentary, multilayered, not fully formalisable knowledge, in which the informal metatheory of the usual formalisation approach is replaced by a semiformal metalogic program which interactively composes formal object theories to be accepted or rejected as formalisations of the knowledge by the user. Our representation easily copes with changes in the represented knowledge.

Imprecise knowledge requires advanced *user interaction* that promotes meaningful user answers and queries, constructs and intelligibly displays proof terms explaining derived conclusions, and makes the system pose its questions in a natural order. These aspects have been considered and to some extent solved in our program [Hamfelt and Hansson 1991b].

Multiple semantic interpretations of provisions is realised by allowing the user to fill schemata with meaningful content referring to his fact situation whereupon the system accepts or rejects the thus proposed rule. Including multiple structural interpretations, e.g., adding premises, should raise no real obstacles provided rules of acceptance for such alteration can be established.

In *case law* rules of legal interpretation are as important as in statute law and apart from the difficult problem of inducing schemata from precedent cases, we hypothesize that our framework needs only minor adaptations to catch the problem of case-based reasoning.

Proof terms should, since the notion of being a persuasive line of arguments is vague, not only be displayed for user communication but also reasoned about.

Acknowledgments

We like to thank Keith Clark and Leon Sterling for valuable comments.

References

[Allen and Saxon 1991] L. E. Allen and S. S. Saxon. More IA Needed in AI: Interpretation Assistance for Coping with the Problem of Multiple Structural Interpretations. In *Proc. Third Int. Conf. on Artificial Intelligence and Law*, ACM, New York, 1991. pp. 53-61.

[Bench-Capon and Sergot 1988] T. Bench-Capon and M. Sergot. Toward a Rule-Based Representation of Open Texture in Law. *Computer Power and Legal Language*, ed. C. Walter, Quorum Books, New York, 1988. pp. 39–60.

[Bowen and Kowalski 1982] K. A. Bowen and R. A. Kowalski. Amalgamating Language and Metalanguage in Logic Programming. *Logic Programming*, eds. K. Clark and S.-Å. Tärnlund, Academic Press, London, 1982. pp. 153–72.

[Bratley *et al.* 1991] P. Bratley, J. Fremont, E. Mackaay and D. Poulin. Coping with Change. In *Proc. Third Int. Conf. on Artificial Intelligence and Law*, ACM, New York, 1991. pp. 69–75.

[Costantini 1990] S. Costantini. Semantics of a Metalogic Programming Language. In *Proc. Second Workshop on Metaprogramming in Logic*, ed. M. Bruynooghe, Katholieke Universiteit Leuven, 1990. pp. 3–18.

[Hamfelt 1990] A. Hamfelt. *The Multilevel Structure of Legal Knowledge and its Representation*, Uppsala Theses in Computing Science 8/90, Uppsala University, Uppsala, 1990.

[Hamfelt and Barklund 1990] A. Hamfelt, J. Barklund. Metaprogramming for Representation of Legal Principles. In *Proc. Second Workshop on Metaprogramming in Logic*, ed. M. Bruynooghe, Katholieke Universiteit Leuven, 1990. pp. 105–22.

[Hamfelt and Hansson 1991a] A. Hamfelt, Å. Hansson. Metalogic Representation of Stratified Knowledge. *UPMAIL TR 66*, Comp. Sci. Dept., Uppsala University, Uppsala, 1991.

[Hamfelt and Hansson 1991b] A. Hamfelt, Å. Hansson. Representation of Fragmentary and Multilayered Knowledge—A Semiformal Metatheory as an Interactive Metalogic Program. *UPMAIL TR 68*, Comp. Sci. Dept., Uppsala University, Uppsala, 1991.

[Horovitz 1972] J. Horovitz. *Law and Logic*. Springer-Verlag, Vienna, 1972.

[Kleene 1980] S. C. Kleene, *Introduction to Metamathematics*. North Holland, New York, 1980.

[Kowalski 1990] R. A. Kowalski. Problems and Promises of Computational Logic. *Computational Logic*, ed. J. W. Lloyd, Springer-Verlag, Berlin, 1990. pp. 1–36.

[Kowalski and Sergot 1990] R. A. Kowalski, M. J. Sergot. The Use of Logical Models in Legal Problem Solving. *Ratio Juris*, Vol. 3, No. 2 (1990), pp. 201–18.

[Sergot *et al.* 1986] M. J. Sergot, F. Sadri, R. A. Kowalski, F. Kriwaczek, P. Hammond, H. T. Cory. The British Nationality Act as a Logic Program. *Comm. ACM* 29, (May 1986), pp. 370–86.

[Sergot 1983] M. J. Sergot. A Query-the-User Facility for Logic Programming. *Integrated Interactive Computer Systems*, eds. P. Degano and E. Sandewall, North-Holland, Amsterdam, 1983. pp. 27–41.