Formalizing Belief Revision in Type Theory

Tijn Borghuis, *Mathematics and Computing Science, Technische Universiteit Eindhoven, P.O.Box 513, 5600 MB Eindhoven, the Netherlands,* v.a.j.borghuis@win.tue.nl

Fairouz Kamareddine, *Mathematical and Computer Sciences, Heriot-Watt Univ., Riccarton, Edinburgh EH14 4AS, Scotland,* fairouz@macs.hw.ac.uk

Rob Nederpelt, *Same address as Borghuis,* r.p.nederpelt@tue.nl

Abstract

This paper formalizes belief revision for belief states in type theory. Type theory has been influential in logic and computer science but as far as we know, this is the first account at using type theory in belief revision. The use of type theory allows an agent's beliefs as well as his justifications for these beliefs to be explicitly represented and hence to act as first-class citizens. Treating justifications as first-class citizens allows for a deductive perspective on belief revision. We propose a procedure for identifying and removing "suspect" beliefs, and beliefs depending on them. The procedure may be applied iteratively, and terminates in a consistent belief state. The procedure is based on introducing explicit justification of beliefs. We study the belief change operations emerging from this perspective in the setting of typed λ-calculus, and situate these operations with respect to standard approaches.

Keywords: Belief Revision, Type Theory, Explicit Justifications, Propositions as Types.

1 Introduction

An agent who keeps expanding his belief state with new information may reach a stage where his beliefs have become inconsistent, and his belief state has to be adapted to regain consistency. Usually, in studying this problem of "belief revision", the justifications an agent has for his beliefs are not considered to be first-class citizens.

The two main approaches in the belief revision literature regarding justifications [17] are:

1. "Foundations theory", in which one needs to keep track of justifications for one's beliefs; propositions that have no justification should not be accepted as beliefs.
2. "Coherence theory", in which one needs not consider justifications; what matters is how a belief coheres with the other beliefs that are accepted in the present state.

In foundations theory, beliefs are held to be justified by one or several other beliefs (and some beliefs are justified by themselves). However, in this view, justifications are only *implicitly* present as relations between beliefs, rather than as objects in their own right which are *explicitly* represented in the formalisation of belief states and belief change operations. Hence, justifications are not first-class citizens in foundations theory, and not considered at all in coherence theory.

However, experience in the past decades shows that when building automated systems and theorem provers, explicit representation is absolutely necessary. This is the case for example in the theorem prover Automath (for automating mathematics, [31]) where definitions (the heart of mathematics) are made explicit. This is also the case in the implementation of programming languages where contexts and environments are made explicit. It turns out also that treating justifications explicitly (hence as first-class citizens), allows for a deductive perspective of belief revision which can be automated.

In this paper, we explore belief revision for belief states in which justifications are first-class citizens represented explicitly. Our motivation for investigating belief revision along these lines stems from working on knowledge representation in Pure Type Systems [4] in the DenK-project [9]. Type theory was chosen due to its excellent success in the field of theorem proving (Automath [31] and Coq [5]) and programming languages (ML [30]). See also [6] where type theory has been shown to be useful for knowledge representation. In the DenK-project a formal model was made of a specific communication situation, and used to implement a human-computer interface. Both in the model and in the system, belief states of agents were formalised as type theoretical contexts. This means that an agent's beliefs are represented in a binary format, where one part is the proposition believed by the agent and the other the justification the agent has for this belief. Both parts are syntactic objects in their own right, and can be calculated upon by means of the rules of the type theory. This way of representing beliefs turns justifications into first-class citizens, and proved to be very fruitful for the purposes of the project.

At that time mechanisms for belief revision were not investigated or implemented, but it became clear that given this formalisation of belief states there is a straightforward deductive approach to the problem: since every belief is accompanied by its justification (and the rules of the calculus operate on both), every inconsistency that surfaces in the agent's belief state has its own (complex) justification containing the justifications of the beliefs that together cause the inconsistency. This makes it easy to identify and remove the "suspects" among the beliefs in the agent's belief state. Although, technically speaking, this is a direct consequence of the so-called Propositions As Types-principle (cf. sections 3 and 4), this simple idea seems not to have been explored before. We feel that this is of a more general interest for two reasons:

1. Our type theoretical case study shows that explicitly represented justifications have clear advantages: a number of drawbacks traditionally associated with foundational approaches disappear. As such, it may serve as a precursor to a more general account in the setting of Labelled Deductive Systems [15], of which typed λ-calculi are a simple case.[1]
2. It may contribute to a more computational account of belief revision, one which is applicable to agents that have finite information and finite reasoning powers.

In developing the idea, we will come across other well-known issues in this field of research. For instance the question whether belief states should be taken to be logically closed sets or rather a base set of beliefs which is not closed under logical

[1] Note that in the conclusion, [16] discusses the possibility of a general theory of inconsistency where an account of belief revision would fall out as a special case. However, as far as we know, this general theory of inconsistency in LDS has not yet been materialized.

consequence [19], and the question whether an agent should always accept new information (prioritized versus non-prioritized revision [21]). In addition, we question a number of assumptions that are traditionally made such as the assumption that an agent has infinite reasoning powers, and that an agent has to solve the revision problem "in splendid isolation", i.e. without going back to his sources of information via observation and communication.

The paper is structured as follows: in Section 2 we review type theory and its untyped basis (the type-free λ-calculus), the propositions-as-types principle and introduce the extension of type theory with definitions that will be used for belief revision. In Section 3, we explain how belief states can be captured in type theory. Section 4 shows how type theoretical belief states develop as new information becomes available, and gives an informal statement of the problem of revision in type theory. This account of type theoretical revision is formalised in Section 5. In Sections 6 and 7 we situate our approach with respect to standard approaches from the literature, and make a comparison on the level of belief change operations. As it turns out, our revision procedure is particularly close to the so-called consolidation operations. This is shown in Appendix A. We conclude in Section 8.

2 Type theory

2.1 Informal introduction

Judgements

The basic relation in type theory is the *judgement*

$$\Gamma \vdash a : T$$

which can be read as 'term a has type T in context Γ'. Here 'a' and 'T' are both formulas written according to a well-defined syntax (on the basis of λ-calculus). The expression $a : T$ is called a *statement*, term a is *the subject* of the statement. One also says that term a is an *inhabitant* of type T.

The context Γ is a list of statements with *variables* as subjects, e.g. $x_1 : T_1, \ldots, x_n : T_n$. The above judgement can then be read as follows: "If x_1 has type T_1, ..., and x_n has type T_n, then term a has type T". Note that a may *contain* x_1, ..., x_n, so a *depends on* x_1 to x_n. The set $\{x_1, \ldots, x_n\}$ is called the *domain* of Γ, or $\mathtt{dom}(\Gamma)$.

Statements

The intuitive notion 'has type' has a direct counterpart in naive set theory, viz. 'is element of'. For example, consider the statement '$a : \mathbf{N}$' ('term a *has type* $\mathbf{N}$'). Assuming that $\mathbf{N}$ is a symbol representing the set of natural numbers, this statement can be interpreted as '$a \in \mathbf{N}$' ('the object represented by a *is element of* the naturals').

The notion of having a type, however, is more general than the notion of set-theoretical elementhood. This is because a type T can represent not only some kind of set, but also *a proposition*. In the latter representation, the statement $a : T$ expresses: 'a is (a term representing) a *proof* of the proposition T'. One speaks of 'propositions as types and proofs as terms' (together abbreviated as *PAT*) in order to emphasize this special usage of types. Section 2.2 below gives more details.

The advantage of PAT is that proofs belong to the object language, not the meta-language. That is, proofs are 'first class citizens' in the syntactical world of type theory. This, combined with the strength of the standard λ-calculus operations, makes type theory a powerful mechanism.

Contexts

The context Γ in a judgement $\Gamma \vdash a : T$ contains the 'prerequisites' necessary for establishing the statement $a : T$. A context Γ is a list of statements with *distinct* variables as subjects, like $x_1 : T_1, \ldots, x_n : T_n$. A context statement $x_i : T_i$ can express several kinds of prerequisites, the simplest being:

1. x_i is an element of the set T_i,
2. T_i is an assumption (a proposition) and x_i is its atomic justification.

However, in type theory there are different 'levels' of typing: a type can have a type itself. Statements expressing the typing of types are concerned with the well-formedness of these types. For the T_i occurring in 1. and 2. above, such statements have the form:

1. $T_i : \texttt{set}$, to express that T_i is a well-formed formula representing a set,
2. $T_i : \texttt{prop}$, to express that T_i is a well-formed formula representing a proposition.

The last-mentioned statements can also be part of a context in the special case that T_1 and T_2 are variables. So a context could look like: $T_1 : \texttt{prop}, T_2 : \texttt{set}, x_1 : T_1, x_2 : T_2$ (to be read as: "let T_1 be a proposition, T_2 a set, x_1 a justification for T_1 and x_2 for T_2"). The terms `set` and `prop` are examples of so-called *sorts*, predefined constants on which the type system is based. Every type system has a specific set of sorts, which we denote by $\mathcal{S}$.

Note that the statements in the context are *ordered*: first arbitrary set T_1 and proposition T_2 are proposed, before their inhabitants x_1 and x_2 are introduced. This is a general principle in contexts: every variable (except the sorts) used in a type must be introduced as the subject of a preceding statement. As a matter of fact, a similar consideration applies to *judgements*: in $\Gamma \vdash a : T$ all variables and (free) constants used in a *and* T must be introduced as subjects in Γ.

2.2 PAT: Propositions As Types

The idea of PAT originates in the formulation of intuitionistic logic where frequently occurring constructions in intuitionistic mathematics have a logical counterpart. One of these constructions is the proof of an implication. Heyting [24] describes the proof of an implication $a \Rightarrow b$ as: Deriving a solution for the problem b from the problem a. Kolmogorov [28] is even more explicit, and describes a proof of $a \Rightarrow b$ as the construction of a method that transforms each proof of a into a proof of b. This means that a proof of $a \Rightarrow b$ can be seen as a *(constructive) function* from the proofs of a to the proofs of b. In other words, the proofs of the proposition $a \Rightarrow b$ form exactly the set of functions from the set of proofs of a to the set of proofs of b. This suggests to identify a proposition with the set of its proofs. Now *types* are used to represent these sets of proofs. An element of such a set of proofs is represented as a *term* of the corresponding type. This means that propositions are interpreted as *types*, and proofs of a proposition a as *terms of type* a.

PAT was, independently from Heyting and Kolmogorov, discovered by Curry and Feys [13]. Howard [25] follows the argument of Curry and Feys [13] and combines it with Tait's discovery of the correspondence between cut elimination and β-reduction of λ-terms [32]. Howard's discovery dates from 1969, but was not published until 1980. Independently of Curry and Feys and Howard, we find a variant of PAT in AUT-68, the first Automath system of De Bruijn [31]. Though De Bruijn was probably influenced by Heyting, his ideas arose independently from Curry, Feys and Howard. This can be clearly seen in Section 2.4 of [8], where propositions as types (or better: Proofs as terms) was implemented in a different way to that of Curry and Howard.

The Propositions as Types and Proofs as Terms (PAT) principle has opened the possibility to use Type Theory not only as a restrictive method (to prevent paradoxes) but also as a constructive method. Many proof checkers and theorem provers, like Automath [31], Coq [5] and LF [23], use the PAT principle (see [29] for more details).

"Proofs as terms" already suggests an important advantage of using type theory as a logical system: In this method proofs are first-class citizens of the logical system, whilst for many other logical systems, proofs are rather complex objects outside the logic (for example: derivation trees), and therefore cannot be easily manipulated.

The fact that PAT was discovered independently by many different people, and its use in various logical frameworks and theorem provers, is an evidence to the usefulness of such notion in logic and compoutation. For our purpose of belief revision, PAT allows to store the developmental history of the justifications of a belief and hence, to retrace back this histoy and to restore inconsistent belief states.

2.3 Theories

A 'proof' is generally considered to be a mathematical notion, but in the PAT-style a proof is anything *justifying* a proposition. This can be a proof in the mathematical sense, but also any other acceptable justification. Let T represent a proposition and let $a : T$. Then:

- If a is an *atomic term* (think of a constant or a variable), then a encodes a justification which cannot be further analysed:
  - It can stand for an axiomatic justification of a proposition: T is an *axiom* and a expresses that the axiom 'holds'.
  - The validity of proposition T can also come from a *reliable source*. In this case the proof a itself cannot be inspected, but the reliability of the source is enough guarantee to accept the proof. The origin of the knowledge can be any source, either virtual: e.g. a knowledge base, or real: a reliable (community of) person(s).
  - Proposition T can also be justified by *observational evidence*. For example, the proposition that a certain body is yellow can be justified by an atomic term representing the observation that this is the case.
  - Finally, proposition T can be an *assumption*. This case is dealt with in type theory by introducing a variable (say x) as an arbitrary (but fresh) inhabitant for the proposition: the statement $x : T$ then expresses: 'Let x be a proof of T'.

Since x is an unspecified variable, this amounts to: 'Assume T' (albeit that the proof x can be called upon later).

- If a is a *composite term*, composed according to the (type-theoretical) syntax, it embodies a complex justification. In this case the precize structure of a expresses how the evidence for T is constructed. For example, under the PAT-interpretation a complete mathematical proof (of a theorem) is coded in one, possibly large, composite term. But also a justification that combines knowledge obtained from observing a certain object with general rules about its behaviour, will lead to a composite term.

The PAT-interpretation enables a well-established connection between mathematics and type theory, as has been shown already in the Automath project [31], in which large parts of mathematics have been formalized in type theory: an entire mathematical theory was rendered as a list of judgements. The great importance of such a type-theoretical formalization is that it makes it possible to check whether a given proof of a certain theorem does indeed *prove* the theorem. In fact, it turns out that *syntactical correctness* of the list of judgements is enough to establish the mathematical correctness of the mathematical theory. And the check on syntactical correctness is relatively easy, since the question whether a certain term is of a certain type in a certain context is decidable. This check on syntactical correctness can be performed by man, but also by a straightforward computer program. In the Automath project, this has already been done with the computer technology of the seventies.

A second advantage is the long-standing connection between logic and type theory. The 'reasoning power' of logic finds a natural counterpart in the operations of λ-calculus underlying type theory. A well-known result is that logics of arbitrarily high order can be expressed in type theory. In the PAT-interpretation of logic, terms capture the full proof *process*: from a proof term one can reconstruct not only the premisses used in the proof, but also the order in which they were used and the logical rules used to combine them.

2.4 The type free λ-calculus

Modern type theory is based on the λ-calculus. This section intrtoduces the type free λ-calculus.

Definition 2.1 (Syntax of λ-terms)
The set of classical λ-terms or λ-expressions $\mathcal{M}$ is given by: $\mathcal{M} ::= \mathcal{V}|(\lambda\mathcal{V}.\mathcal{M})|(\mathcal{M}\mathcal{M})$ where $\mathcal{V} = \{x, y, z, \dots\}$ is an infinite set of *term variables*. We let $v, v', v'', \cdots$ range over $\mathcal{V}$ and $A, B, C \cdots$ range over $\mathcal{M}$.

Example 2.2
$(\lambda x.x)$, $(\lambda x.(xx))$, $(\lambda x.(\lambda y.x))$, $(\lambda x.(\lambda y.(xy)))$, and $((\lambda x.x)(\lambda x.x))$ are all classical λ-expressions.

This simple language is surprisingly rich. Its richness comes from the freedom to create and apply functions, especially higher order functions to other functions (and even to themselves). To explain the intuitive meaning of these three sorts of expressions, let us imagine a model where every λ-expression denotes an element of that model (which is a function). In particular, the variables denote a function in the model via an interpretation function or an *environment* which maps every variable into a specific element of the model. Such a model by the way was not obvious for more than forty years. In fact, for a domain D to be a model of λ-calculus, it requires that the set of functions from D to D be included in D. Moreover, as the λ-calculus represents precisely the recursive functions, we know from Cantor's theorem that the domain D is much smaller than the set of functions from D to D. Dana Scott was armed by this theorem in his attempt to show the non-existence of the models of the λ-calculus. To his surprise, he proved the opposite of what he set out to show. He found in 1969 a model which has opened the door to an extensive area of research in computer science. We will not go into the details of these models in this paper.

DEFINITION 2.3 (Meaning of Terms)
Here is now the intuitive meaning of each of the three λ-expressions given in the syntax:

Variables Functions denoted by variables are determined by what the variables are bound to in the *environment*. Binding is done by λ-abstraction.

Function application If A and B are λ-expressions, then so is (AB). This expression denotes the result of applying the function denoted by A to the function denoted by B.

Abstraction If v is a variable and A is an expression which may or may not contain occurrences of v, then $\lambda v.A$ denotes the function that maps the input value B to the output value $A[v := B]$, that is: the expression A in which B has been substituted for v.

EXAMPLE 2.4
$(\lambda x.x)$ denotes the identity function. $(\lambda x.(\lambda y.x))$ denotes the function which takes two arguments and returns the first.

As parentheses are cumbersome, we will use the following notational convention:

DEFINITION 2.5 (Notational convention)
We use these notational conventions:

1. Functional application associates to the left. So ABC denotes $((AB)C)$.
2. The body of a λ is anything that comes after it. So, instead of $(\lambda v.(A_1A_2 \ldots A_n))$, we write $\lambda v.A_1A_2 \ldots A_n$.
3. A sequence of λ's is compressed to one, so $\lambda xyz.t$ denotes $\lambda x.(\lambda y.(\lambda z.t))$.

As a consequence of these notational conventions we get:

1. Parentheses may be dropped: (AB) and $(\lambda v.A)$ are written AB and $\lambda v.A$.
2. Application has priority over abstraction: $\lambda x.yz$ means $\lambda x.(yz)$ and not $(\lambda x.y)z$.

2.4.1 Variables and Substitution

We need to manipulate λ-expressions in order to get values. For example, we need to apply $(\lambda x.x)$ to y to obtain y. To do so, we use the β-rule which says that $(\lambda v.A)B$ evaluates to *the body* A where v is substituted by B, written $A[v := B]$. However, one has to be careful. Look at the following example:

Example 2.6
Evaluating $(\lambda fx.fx)g$ to $\lambda x.gx$ is perfectly acceptable but evaluating $(\lambda fx.fx)x$ to $\lambda x.xx$ is not. By Definition 2.3, $\lambda fx.fx$ and $\lambda fy.fy$ have the same meaning and hence $(\lambda fx.fx)x$ and $(\lambda fy.fy)x$ must also have the same meaning. Moreover, their values must have the same meaning. However, if $(\lambda fx.fx)x$ evaluates to $\lambda x.xx$ and $(\lambda fy.fy)x$ evaluates to $\lambda y.xy$, then we easily see, according to Definition 2.3, that $\lambda x.xx$ and $\lambda y.xy$ have two different meanings. The first takes a function and applies it to itself, the second takes a function y and applies x (whatever its value) to y.

We define the notions of *free* and *bound* variables which will play an important role in avoiding the problem above. In fact, the λ is a variable binder, just like $\forall$ in logic:

Definition 2.7 (Free and Bound variables)
For a λ-term C, the set of free variables $FV(C)$, and the set of bound variables $BV(C)$, are defined inductively as follows:

$$\begin{array}{lll@{\qquad}lll}
FV(v) & =_{def} & \{v\} & BV(v) & =_{def} & \emptyset \\
FV(\lambda v.A) & =_{def} & FV(A) - \{v\} & BV(\lambda v.A) & =_{def} & BV(A) \cup \{v\} \\
FV(AB) & =_{def} & FV(A) \cup FV(B) & BV(AB) & =_{def} & BV(A) \cup BV(B)
\end{array}$$

An occurrence of a variable v in a λ-expression is free if it is not within the scope of a λv.[2], otherwise it is bound. For example, in $(\lambda x.yx)(\lambda y.xy)$, the first occurrence of y is free whereas the second is bound. Moreover, the first occurrence of x is bound whereas the second is free. In $\lambda y.x(\lambda x.yx)$ the first occurrence of x is free whereas the second is bound. A *closed term* is a λ-term in which all variables are bound.

Here is now the definition of substitution:

Definition 2.8 (Substitution)
For any A, B, v, we define $A[v := B]$ to be the result of substituting B for every free occurrence of v in A, as follows:

$$\begin{array}{lcl}
v[v := B] & \equiv & B \\
v'[v := B] & \equiv & v \qquad\qquad \text{if } v \not\equiv v' \\
(AC)[v := B] & \equiv & A[v := B]C[v := B] \\
(\lambda v.A)[v := B] & \equiv & \lambda v.A \\
(\lambda v'.A)[v := B] & \equiv & \lambda v'.A[v := B] \\
 & & \text{if } v' \not\equiv v \text{ and } (v' \notin FV(B) \text{ or } v \notin FV(A)) \\
(\lambda v'.A)[v := B] & \equiv & \lambda v''.A[v' := v''][v := B] \\
 & & \text{if } v' \not\equiv v \text{ and } (v' \in FV(B) \text{ and } v \in FV(A))
\end{array}$$

In the last clause, v'' is chosen to be the first variable $\notin FV(AB)$. In the case when terms are identified modulo the names of their bound variables, then in the last clause of the above definition, any $v'' \notin FV(AB)$ can be taken. In implementation however, this identification is useless and a particular choice of v'' has to be made.

Example 2.9
Check that $(\lambda y.yx)[x := z] \equiv \lambda y.yz$, that $(\lambda y.yx)[x := y] \equiv \lambda z.zy$, and that $(\lambda y.yz)[x := \lambda z.z] \equiv \lambda y.yz$.

Lemma 2.10 (Substitution for variable names)
Let $A, B, C \in \mathcal{M}$, x, y, $\in \mathcal{V}$. For $x \neq y$ and $x \notin \mathtt{FV}(C)$, we have that: $A[x := B][y := C] \equiv A[y := C][x := B[y := C]]$.

[2]Notice that the v in λv is not an occurrence of v.

2.4.2 Reduction

The two important notions of reduction are α-reduction which identifies terms up to variable renaming and β-reduction which evaluates λ-terms.

DEFINITION 2.11 (Compatibility for the type free λ-calculus)
We say that a binary relation $\rightarrow$ on the type free λ-calculus is compatible iff for all terms A, B of the λ-calculus and variable v, the following holds:

$$\frac{A \rightarrow B}{AC \rightarrow BC} \qquad \frac{A \rightarrow B}{CA \rightarrow CB} \qquad \frac{A \rightarrow B}{\lambda v.A \rightarrow \lambda v.B}$$

DEFINITION 2.12 (Alpha reduction)
$\rightarrow_\alpha$ is defined to be the least compatible relation closed under the axiom:

$$(\alpha) \qquad \lambda v.A \rightarrow_\alpha \lambda v'.A[v := v'] \qquad \text{where } v' \notin FV(A)$$

EXAMPLE 2.13
$\lambda x.x \rightarrow_\alpha \lambda y.y$ but it is not the case that $\lambda x.xy \rightarrow_\alpha \lambda y.yy$.
Moreover, $\lambda z.(\lambda x.x)x \twoheadrightarrow_\alpha \lambda z.(\lambda y.y)x$.

Recall that $\lambda x.x \not\equiv \lambda y.y$ even though they represent the same function. They are actually identical modulo α-conversion. I.e. $\lambda x.x =_\alpha \lambda y.y$.

DEFINITION 2.14 (Beta reduction)
$\rightarrow_\beta$ is defined to be the least compatible relation closed under the axiom:

$$(\beta) \qquad (\lambda v.A)B \rightarrow_\beta A[v := B]$$

We use $\twoheadrightarrow_\beta$ to denote the reflexive transitive closure of $\rightarrow_\beta$. We say that a term A is a β-normal form if there is no B such that $A \rightarrow_\beta B$.

EXAMPLE 2.15
Check that $(\lambda x.x)(\lambda z.z) \rightarrow_\beta \lambda z.z$, that $(\lambda y.(\lambda x.x)(\lambda z.z))xy \twoheadrightarrow_\beta y$, and that both $\lambda z.z$ and y are β-normal forms.

Here is a lemma about the interaction of β-reduction and substitution:

LEMMA 2.16
Let $A, B, C, D \in \mathcal{M}$.

1. If $C \rightarrow_\beta D$ then $A[x := C] \twoheadrightarrow_\beta A[x := D]$.
2. If $A \rightarrow_\beta B$ then $A[x := C] \rightarrow_\beta B[x := C]$.

PROOF. By induction on the structure of A for 1, on the derivation $A \rightarrow_\beta B$ for 2. ∎

2.5 The syntax and rules of Pure Type Systems

Now we are ready to introduce the syntax and rules of Pure Type Systems (PTSs) which will be the basis of our theory of belief revision. There are two type disciplines: the implicit and the explicit. The implicit style, also known as typing à la Curry, does not annotate variables with types. For example, the identity function is written as in the type-free case, as $\lambda x.x$. The type of terms however is found using the typing rules of the system in use. The explicit style, also known as typing à la Church, does annotate variables and the identity function may be written as $\lambda x : \texttt{Bool}.x$ to represent identity over booleans. In this paper, we consider typing à la Church. We present what is known as *Pure Type Systems* or PTSs. Important type systems that are PTSs include Church's simply typed λ-calculus [11] and the calculus of constructions [12] which are also systems of the Barendregt cube [4]. Berardi [7] and Terlouw [33] have independently generalised the method of generating type systems into the pure type systems framework. This generalisation has many advantages. First, it enables one to introduce eight logical systems that are in close correspondence with the systems of the Barendregt cube. Those eight logical systems can each be described as a PTS in such a way that the propositions-as-types interpretation obtains a canonical system form [4]. Second, the general setting of the PTSs makes it easier to write various proofs about the systems of the cube.

In PTSs, we have in addition to the usual λ-abstraction, a type forming operator Π. Briefly, if A is a type, and B is a type possibly containing the variable x, then $\Pi x{:}A.B$ is the type of functions that, given a term $a : A$, output a value of type $B[x := a]$. Here, again, $a : A$ expresses that a is of type A. If x does not occur in B, then $\Pi x{:}A.B$ is the type of functions from A to B, written $A \to B$. To the Π-abstraction at the level of types corresponds λ-abstraction at the level of objects. Roughly speaking, if M is a term of type B (M and B possibly containing x), then $\lambda x{:}A.M$ is a term of type $\Pi x{:}A.B$. All PTSs have the same typing rules but are distinguished from one another by the set $\mathcal{R}$ of triples of sorts (s_1, s_2, s_3) allowed in the so-called *type-formation* or Π-*formation* rule, (product). Each PTS has its own set $\mathcal{R}$. A Π-type can only be formed in a specific PTS if the (product) rule is satisfied for some (s_1, s_2, s_3) in the set $\mathcal{R}$ of that system. (see Figure 1).

Definition 2.17
The *set of pseudo-terms* $\mathcal{T}$, is generated by the grammar:
$\mathcal{T} ::= \mathcal{V} \mid \mathcal{C} \mid (\mathcal{T}\,\mathcal{T}) \mid (\lambda\mathcal{V} : \mathcal{T}.\mathcal{T}) \mid (\Pi\mathcal{V} : \mathcal{T}.\mathcal{T})$, where $\mathcal{V}$ is the infinite set of variables $\{x, y, z, \dots\}$ and $\mathcal{C}$ a set of constants over which, $c, c_1, \dots$ range. We use $A, B, \dots$ to range over $\mathcal{T}$ and $v, v', v'', \dots$ to range over $\mathcal{V}$. Throughout, we take $\pi \in \{\lambda, \Pi\}$.

Note that in the type free lambda calculus, there were only three possibilities for terms (given in Definition 2.1): variables, applications or abstractions, and that abstractions contained no typings for the variables abstracted over. The above Definition 2.17 on the other hand, gives the typing of the abstracted variable, and also defines types as well as terms. $\mathcal{C}$ is a set of constants which contains a subset $\mathcal{S}$ called the *sorts.* The set *sorts* contains amongst other things, four special elements: $\texttt{set}$, $\texttt{prop}$, $*$ and $\Box$, with the relations to be defined later that: $\texttt{set}: *$, $\texttt{prop}: *$ and $* : \Box$. If $A : *$ (resp. $A : \Box$) we say that A is a type (resp. a kind). If $A : \texttt{set}$ (resp. $A : \texttt{prop}$), then we consider A as a set (resp. a proposition).

Definition 2.18 (Free and Bound variables)
The free and bound variables in terms are defined similarly to those of Definition 2.7 with the exception that $FV(c) =_{def} BV(c) =_{def} \emptyset$ and in the case of abstraction, $FV(\pi v : A.B) =_{def} (FV(B) \setminus \{v\}) \cup FV(A)$ and $BV(\pi v : A.B) =_{def} BV(A) \cup BV(B) \cup \{v\}$.

We write $A[x := B]$ to denote the term where all the free occurrences of x in A have been replaced by B. Furthermore, we take terms to be equivalent up to variable renaming. We assume moreover, the Barendregt variable convention which is formally stated as follows:

Convention 2.19
(VC: Barendregt's Convention) Names of bound variables will always be chosen such that they differ from the free ones in a term. Moreover, different λ's have different variables as subscript. Hence, we will not have $(\lambda x : A.x)x$, but $(\lambda y : A.y)x$ instead.

The definition of compatibility of a reduction relation for PTSs is that of the type-free calculus (given in Definition 2.11) but where the case of abstraction is replaced by:

$$\frac{A_1 \to A_2}{\pi x : A_1.B \to \pi x : A_2.B} \qquad \frac{B_1 \to B_2}{\pi x : A.B_1 \to \pi x : A.B_2}$$

Definition 2.20
β-reduction is the least compatible relation on $\mathcal{T}$ generated by

$$(\beta) \qquad (\lambda x : A.B)C \to B[x := C]$$

Note that $(\lambda x : A.B)C$ is reduced and not $(\Pi x : A.B)C$. The latter needs special attention as is shown in [26, 27].

Now, we define some machinery needed for typing:

Definition 2.21

1. A *statement* is of the form $A : B$ with $A, B \in \mathcal{T}$. We call A the *subject* and B the *predicate* of $A : B$.
2. A *declaration* is of the form $x : A$ with $A \in \mathcal{T}$ and $x \in \mathcal{V}$. When d is $x : A$, we define $\mathtt{var}(d)$ and $\mathtt{type}(d)$ to be x and A respectively.
3. A *pseudo-context* is a finite ordered sequence of declarations, all with distinct subjects. We use $\Gamma, \Delta, \Gamma', \Gamma_1, \Gamma_2, \ldots$ to range over pseudo-contexts. The *empty* context is denoted by either $<>$ or nothing at all.
4. If $\Gamma = x_1 : A_1. \ldots .x_n : A_n$ then $\Gamma, x : B = x_1 : A_1, \ldots, x_n : A_n, x : B$ and $\mathtt{dom}(\Gamma) = \{x_1, \ldots, x_n\}$.
5. We define substitutions on contexts by: $\emptyset[x := A] \equiv \emptyset$, and $(\Gamma, y : B)[x := A] \equiv \Gamma[x := A], y : B[x := A]$.

Definition 2.22
A *type assignment* relation is a relation between a pseudo-context and two pseudo-terms written as $\Gamma \vdash A : B$. The *rules of type assignment* establish which *judgments* $\Gamma \vdash A : B$ can be derived. A judgement $\Gamma \vdash A : B$ states that $A : B$ can be derived from the pseudo-context Γ.

Definition 2.23
Let Γ be a pseudo-context, A be a pseudo-term and $\vdash$ be a type assignment relation.

1. Γ is called legal if $\exists A, B \in \mathcal{T}$ such that $\Gamma \vdash A : B$.
2. $A \in \mathcal{T}$ is called a Γ-$\mathtt{term}$ if $\exists B \in \mathcal{T}$ such that $\Gamma \vdash A : B$ or $\Gamma \vdash B : A$.
   We take Γ-terms $= \{A \in \mathcal{T}$ such that $\exists B \in \mathcal{T}$ and $\Gamma \vdash A : B \vee \Gamma \vdash B : A\}$.
3. $A \in \mathcal{T}$ is called legal if $\exists \Gamma$ such that $A \in \Gamma$-terms.

$$
\begin{array}{lll}
\text{(axioms)} & \vdash c : s & \text{if } \; c : s \in \mathcal{A} \\[2ex]
\text{(start)} & \dfrac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A} & \text{if } \; x \notin \Gamma \\[2ex]
\text{(weakening)} & \dfrac{\Gamma \vdash B : C \quad \Gamma \vdash A : s}{\Gamma, x : A \vdash B : C} & \text{if } \; x \notin \Gamma \\[2ex]
\text{(product)} & \dfrac{\Gamma \vdash A : s_1 \quad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash (\Pi x : A.B) : s_3} & \text{if } \; (s_1, s_2, s_3) \in \mathcal{R} \\[2ex]
\text{(application)} & \dfrac{\Gamma \vdash F : (\Pi x : A.B) \quad \Gamma \vdash C : A}{\Gamma \vdash F\, C : B[x := C]} & \\[2ex]
\text{(abstraction)} & \dfrac{\Gamma, x : A \vdash C : B \quad \Gamma \vdash (\Pi x : A.B) : s}{\Gamma \vdash (\lambda x : A.C) : (\Pi x : A.B)} & \\[2ex]
\text{(conversion)} & \dfrac{\Gamma \vdash A : B \quad \Gamma \vdash B' : s \quad B =_\beta B'}{\Gamma \vdash A : B'} &
\end{array}
$$

FIG. 1. PTSs with variables names

DEFINITION 2.24
The *specification* of a PTS is a triple $S = (\mathcal{S}, \mathcal{A}, \mathcal{R})$, where $\mathcal{S}$ is a subset of $\mathcal{C}$, called the *sorts*. $\mathcal{A}$ is a set of *axioms* of the form $c : s$ with $c \in \mathcal{C}$ and $s \in \mathcal{S}$ and $\mathcal{R}$ is a set of *rules* of the form (s_1, s_2, s_3) with $s_1, s_2, s_3 \in \mathcal{S}$.

DEFINITION 2.25
The notion of type derivation, denoted $\Gamma \vdash_{\lambda S} A : B$ (or simply $\Gamma \vdash A : B$), in a PTS whose specification is $S = (\mathcal{S}, \mathcal{A}, \mathcal{R})$, is axiomatised by the axioms and rules of Figure 1.

REMARK 2.26
Note that in Figure 1, we insist in the (start) and (weakening) rules that $x \notin \Gamma$, but we do not insist that $x \notin A$. The condition that $x \notin A$ can be derived from from the fact that $x \notin \Gamma$, that $\Gamma \vdash A : s$ and the properties of PTSs.

Each of the eight systems of the cube is obtained by taking $\mathcal{S} = \{*, \Box\}$, $\mathcal{A} = \{*, \Box\}$, and R to be a set of rules of the form (s_1, s_2, s_2) for $s_1, s_2 \in \{*, \Box\}$. We denote rules of the form (s_1, s_2, s_2) by (s_1, s_2). This means that the only possible (s_1, s_2) rules in the set R (in the case of the cube) are elements of the following set: $\{(*, *), (*, \Box), (\Box, *), (\Box, \Box)\}$. The basic system is the one where $(s_1, s_2) = (*, *)$ is the only possible choice. All other systems have this version of the formation rules, plus one or more other combinations of $(*, \Box)$, $(\Box, *)$ and $(\Box, \Box)$ for (s_1, s_2). See Figures 2 and 3.

$\lambda_{\to}$	$(*,*)$			
$\lambda 2$	$(*,*)$	$(\Box,*)$		
λP	$(*,*)$		$(*,\Box)$	
$\lambda P2$	$(*,*)$	$(\Box,*)$	$(*,\Box)$	
$\lambda\underline{\omega}$	$(*,*)$			$(\Box,\Box)$
$\lambda\omega$	$(*,*)$	$(\Box,*)$		$(\Box,\Box)$
$\lambda P\underline{\omega}$	$(*,*)$		$(*,\Box)$	$(\Box,\Box)$
$\lambda P\omega = \lambda C$	$(*,*)$	$(\Box,*)$	$(*,\Box)$	$(\Box,\Box)$

FIG. 2. Different type formation condition

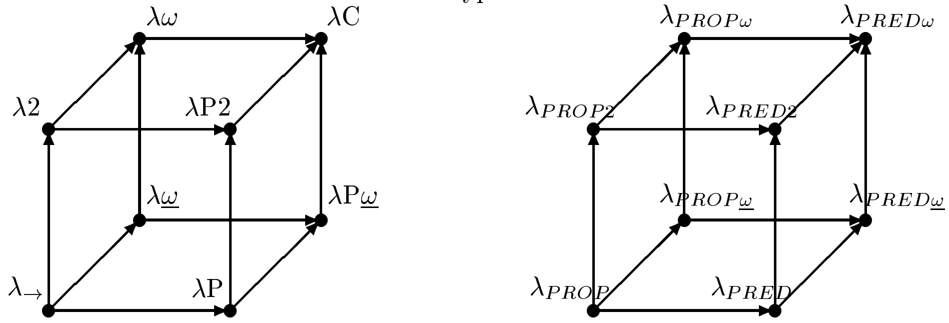

FIG. 3. The λ-cube and its corresponding logic cube

3 Type theory for knowledge representation

This section sets the stage for our account of belief revision with explicit justifications. We give our definition of knowledge and knowledge state, and explain how such knowledge states can be formalized in type theory.

3.1 Knowledge and type theory

PAT is suitable to express the proof as an object *embodying* its developmental history. As a consequence, type theory embodies an excellent machinery for storing (various kinds of) information, including knowledge. The connection between type theory and knowledge is the subject of this section.

We do not intend to present a philosophical or psychological theory of knowledge, but simply identify three characteristics of knowledge which we believe should be taken into account when formalizing knowledge:

- *Subjectivity:* Knowledge is formulated in terms of *concepts*. We assume these concepts are subjective in the sense that one person may judge something to be an instance of a certain concept, while another person would not recognize it as such. Another aspect of subjectivity is that a person's knowledge is *partial*: no one knows everything, and people differ in what they do and don't know.
- *Justification:* Knowledge is justified: persons not only *know* things, but they have *reasons* for knowing them. Generally, parts of knowledge are justified in terms of more basic parts; a person's body of knowledge is structured. And even atomic

justifications are supports for the knowledge, since they point at an origin (an axiom, an observation, etc.).

- *Incrementality:* The knowledge of a person can be *extended* as new information becomes available. Whether this information can be incorporated by the person depends on the possibility to tie it to the knowledge already present. This may lead to simply adding the new information, to dismissing it (e.g., because it is incomprehensible) or even to a reorganization of the existing knowledge.

Under an account of knowledge satisfying these requirements, the traditionally made distinction between knowledge and belief disappears: there can be no knowledge which is true in any absolute sense, since an agent's knowledge depends on his subjective conceptualisation of the world. At best some pieces of knowledge turn out to be more reliable than others and some things can be agreed upon by more agents than others. There is a natural way to capture these characteristics in type theory:

- *Subjectivity is captured by types:* Each concept is formalized as a type, each instance of the concept is a term inhabiting this type. A person's subjective ability to recognize something as an instance of a concept, is mirrored in the ability to judge that the corresponding term inhabits the corresponding type.

  Note that 'having a concept' is also subjective in the sense that different people may have formed different concepts in the course of time. This means that one person can have a concept, whereas another person has no comparable concept. And in case persons *do* have comparable concepts, they may differ in what they recognise as belonging to this concept. In case the type formalizing the concept is a 'set-type', this means that they may differ in what they regard as elements of the set (a rhododendron may be a tree for the one, but a shrub for the other). In case this type is a 'proposition-type', they may differ in what they accept as a justification for that proposition.
- *Justification is captured by terms:* By the PAT-principle, justifications are first-class citizens, formalized in the type-theoretical syntax as terms. The fact that term a justifies proposition T, is expressed as the statement $a : T$. The rules of type theory allow these terms to be combined into complex terms, which reflects that parts of knowledge may be a structured combination of more basic parts.
- *Incrementality is captured by contexts:* As we will explain below, a person's knowledge state can be formalized as a type-theoretical context. Addition of new information to the knowledge state can be formalized by adding statements to the context, dismissing information amounts to reducing the context. Information may only be added if it 'matches' a person's knowledge state. Type theory has an innate notion of 'matching': a statement can only extend a context if it obeys certain well-formedness restrictions.

3.2 Formalization of the knowledge state

The knowledge *state* of a person consists of 'everything he knows' at a certain instant. This knowledge state will be represented as a context Γ in our type system. Every statement in Γ represents a piece of knowledge the person has.

Given our characterization of knowledge, this means that everything in a knowledge state is formulated in terms of the person's concepts. This has several aspects:

- *Meaningfulness:* A person has formed his own, private concepts, and only things which are formulated by means of these concepts can be meaningful to him. Whether or not information coming from outside (by observation or communication) makes sense, depends on the concepts that are already available. (In this paper we will assume that the entirety of concepts of a person is fixed.)
- *Inhabitation:* Whatever a person knows about the world around him is recorded in a knowledge state in the form of meaningful expressions that he accepts. This includes expressions about which objects 'inhabit' the concepts in the world, and which propositions hold in the world, according to the person.

If we take the following (very simple) context as representing a person's knowledge states: $T_1 : \texttt{prop}, T_2 : \texttt{set}, x_1 : T_1, x_2 : T_2$, we can see:

- *Meaningfulness is captured by statements of the form $T : \texttt{prop}$ or $T : \texttt{set}$.* That is to say, in this example the person has two concepts, viz. T_1, which is a proposition to him, and T_2, which is a set. (Note that the statements $T_1 : \texttt{prop}$ by itself does not imply that the proposition T_1 holds according to the person, nor does $T_2 : \texttt{set}$ imply that the set T_2 is non-empty.) At this stage, there are no other concepts, i.e. all sets and propositions which are not constructed out of T_1 and/or T_2 are not meaningful to him.
- *Inhabitation is captured by statements of the form $x : T$, where T is meaningful.* In the example context, the inhabitant x_1 of T_1 represents the person's justification for the holding of T_1, and the inhabitant x_2 of T_2 is an element of the set T_2 which is recognized as such by the person[3].

'Everything a person knows' at a certain instant can be divided into two categories:

- *Explicit knowledge* is expressed by the statements in the context Γ . These are explicitly represented pieces of knowledge directly available to the person.
- *Implicit knowledge* is expressed by statements *derivable* on the context Γ. These are consequences of a person's explicit knowledge which he can get by inference.

Hence, in a judgement of the form $\Gamma \vdash a : T$, the explicit knowledge can be found to the left of the symbol $\vdash$, and the implicit knowledge to the right of $\vdash$.

Note that the knowledge state is not deductively closed, i.e. deriving consequences requires 'work', which is reflected in the construction of a compound justification a for T. Such a construction is a derivation using the rules of type theory; it consists of a *sequence* of judgements of which the just-mentioned compound justification is the final one. We come back to this in the next section.

Assumption 3.1
In order to derive *all* consequences of his explicit knowledge, a person would have to be able to perform possibly infinite derivations. Since this is not feasible, we assume a 'bound' on the derivation depth.

As the above discussion meant that statements of the form $A : B$ (where A may be complex) must be in the knowledge state (which is a context in type theory), and as formulations of type theory only allow statements of the form $x : B$ in the context,

[3]Syntactically, x_1 and x_2 are *variables*. However, as we see later, each of these 'variables' may in fact be a *defined constant*, abbreviating a term which codes all details of the justification.

we will present here an extension of type theory where contexts not only contain statements of the form $x; B$, but also statements of the form $x := A : B$ (which also states that $A : B$), and are known as definitions.

3.3 PTSs with definitions

In this section we introduce the extension of PTSs given in section 2.5 with definitions.

Terms and types remain unchanged, but contexts are now a list of declarations of the form $x : A$ or of *definitions* of the form $x = B : A$. These latter definitions define x to be B and to have the type A. We extend Definition 2.21 to deal with definitions as well as declarations, taking $\mathtt{var}(d)$, $\mathtt{type}(d)$ and $\mathtt{def}(d)$ to be x, A, and B respectively when d is $x = B : A$. We define $FV(x = B : A) \equiv FV(A) \cup FV(B)$. We extend $\mathtt{dom}$ to be $\mathtt{dom}(\Gamma) = \{x \mid x : A \in \Gamma \text{ or } x := B : A \in \Gamma\}$. Finally, we extend substitutions on contexts by $(\Gamma, y := B : C)[x := A] \equiv \Gamma[x := A], y := B[x := A] : C[x := A]$. Note that Definitions 2.22, 2.23, 2.24 and 2.25 are unchanged.

Definition 3.2
The new typing relation $\vdash$ is obtained by adding four new rules to the typing rules of Definition 2.25: (start-def), (weak-def), and (def) below, and by replacing the (conversion) by (new-conv) as follows:

$$\text{(start-def)} \quad \frac{\Gamma \vdash A : s \qquad \Gamma \vdash B : A}{\Gamma, x := B{:}A \vdash x : A} \quad x \notin \mathtt{dom}(\Gamma)$$

$$\text{(weak-def)} \quad \frac{\Gamma \vdash A : B \qquad \Gamma \vdash C : s \quad \Gamma \vdash D : C}{\Gamma, x := D{:}C \vdash A : B} \quad x \notin \mathtt{dom}(\Gamma)$$

$$\text{(def)} \quad \frac{\Gamma, x := B{:}A \vdash C : D}{\Gamma \vdash (\pi x : A.C)B : D[x := B]} \quad \text{for } \pi \in \{\lambda, \Pi\}$$

$$\text{(new-conv)} \quad \frac{\Gamma \vdash A : B \qquad \Gamma \vdash B' : s \qquad \Gamma \vdash B =_{def} B'}{\Gamma \vdash A : B'}$$

In (new-conv), $\Gamma \vdash B =_{def} B'$ is defined as the smallest equivalence relation closed under:

- If $B =_\beta B'$ then $\Gamma \vdash B =_{def} B'$
- If $x := D : C \in \Gamma$ and B' arises from B by substituting one particular free occurrence of x in B by D then $\Gamma \vdash B =_{def} B'$.

In Definition 3.2, (start-def) and (weak-def) are the start and weakening rules that deal with definitions in the context. The (def) rule types λ- and Π-redexes using definitions in the context.

Now, here are some lemmas that show that the above system is suitable for representing beliefs. The first lemma establishes that different beliefs have different justifications and that all justifications have their evidence in knowledge state Γ.

Lemma 3.3 (Free variable Lemma for $\vdash$)
1. If d and d' are two different elements in a legal context Γ, then $\mathtt{var}(d) \not\equiv \mathtt{var}(d')$.
2. If $\Gamma \equiv \Gamma_1, d, \Gamma_2$ and $\Gamma \vdash B : C$ then $FV(d) \subseteq \mathtt{dom}(\Gamma_1)$ and $FV(B), FV(C) \subseteq \mathtt{dom}(\Gamma)$.

PROOF. 1. If Γ is legal then for some B, C, $\Gamma \vdash B : C$. Now use induction on the derivation of $\Gamma \vdash B : C$. 2. is by induction on the derivation of $\Gamma \vdash B : C$. ∎

LEMMA 3.4 (Substitution Lemma for $\vdash$)
If $\Gamma, x := D : C, \Delta \vdash A : B$ or ($\Gamma, x : C, \Delta \vdash A : B$ and $\Gamma \vdash D : C$) then $\Gamma, \Delta[x := D] \vdash A[x := D] : B[x := D]$.

PROOF. Induction on the derivation rules, using Lemma 3.3. ∎

The following corollary means that the person can track down those statements responsible for him entertaining a particular belief.

COROLLARY 3.5 (Strengthening Lemma for $\vdash$)
For $\Gamma_1, y := E : T, \Gamma_2$ a legal context and M and B terms: if $\Gamma_1, y := E : T, \Gamma_2 \vdash M : B$ and $y \notin FV(\Gamma_2) \cup FV(M) \cup FV(B)$, then $\Gamma_1, \Gamma_2 \vdash M : B$.

The next lemma shows that all statements in a knowledge state are meaningful in the sense that if $\Gamma_1, x : A, \Gamma_2$ is legal then $\Gamma_1 \vdash x : A$; and if $\Gamma_1, x := B : A, \Gamma_2$ is legal then $\Gamma_1 \vdash x : A$ and $\Gamma_1 \vdash B : A$

LEMMA 3.6 (Context Lemma for $\vdash$)
Let Γ_1, d, Γ_2 be a legal context. Then we have: $\Gamma_1 \vdash \mathtt{type}(d) : s$ for some sort s,
$\Gamma_1, d \vdash \mathtt{var}(d) : \mathtt{type}(d)$ and if d is a definition then $\Gamma_1 \vdash \mathtt{def}(d) : \mathtt{type}(d)$.

PROOF. If Γ is legal then for some terms B, C: $\Gamma \vdash B : C$; now use induction on the derivation of $\Gamma \vdash B : C$. ∎

LEMMA 3.7 (Thinning Lemma for $\vdash$)
Let d be either a declaration or a definition and let Γ_1, d, Γ_2 be a legal context.

1. If $\Gamma_1, \Gamma_2 \vdash A : B$, then $\Gamma_1, d, \Gamma_2 \vdash A : B$.
2. If d is $x := D : C$ and $\Gamma_1, x : C, \Gamma_2 \vdash A : B$, then $\Gamma_1, d, \Gamma_2 \vdash A : B$.

LEMMA 3.8 (Swap Lemma for $\vdash$)
Assume each of d_1 and d_2 is either a declaration or a definition such that $\mathtt{var}(d_1) \notin FV(\mathtt{type}(d_2))$ and if d_2 is a definition then also $\mathtt{var}(d_1) \notin FV(\mathtt{def}(d_2))$.
If $\Gamma_1, d_1, d_2, \Gamma_2 \vdash A : B$, then $\Gamma_1, d_2, d_1, \Gamma_2 \vdash A : B$.

PROOF. By induction on the derivation $\Gamma_1, d_1, d_2, \Gamma_2 \vdash A : B$. ∎

4 Development of the knowledge state

The knowledge state of a person is not static. As time goes by, new information comes to the person's attention and has to be dealt with. With the conception of knowledge states as type-theoretical contexts in mind, as explained in the previous section, we distinguish several stages in the treatment of new information by a person, marked by decisions which the person has to make. We describe these stages below.

Meaningfulness In the first stage, the meaningfulness of the new information is at stake. New information may or may not be meaningful to a person depending on his current knowledge state. Type-theoretically, new information manifests itself in the form of a (sequence of) statement(s). Whether these statements are meaningful with respect to a knowledge state, can be syntactically decided. In section 3.2 we noted that type theory has an intrinsic notion of meaningfulness. Below we explain how this notion can be extended to statements of the form $x : T$, expressing the inhabitation of a proposition or set T.

We presuppose that a person only processes new information that is meaningful (makes sense) to him, i.e. meaningful with respect to his current knowledge state, and that he *decides* to dismiss this information otherwise. (In a communication setting, we expect the person to search for clarification, either by questioning his dialogue partner, or by (re-)inspecting his environment.)

Expanding the knowledge state If the information *is* meaningful, the person adds it provisionally to the knowledge state: Γ is extended to e.g. $\Gamma_1 \equiv \Gamma, y_1 : T_1, y_2 : T_2$.

The resulting knowledge state can turn out to be consistent, that is to say, the person cannot construct a term M such that $\Gamma_1 \vdash M : \bot$, where $\bot$ is falsum (the logical constant 'falsity'). Recall assumption 3.1 where we assume that the person has a limited deductive power, so he can only construct terms by derivations up to a certain length. Intuitively this means that the person has a 'horizon' behind which he cannot see the consequences of his knowledge state. Hence, the person's notion of 'consistency' is bound by his horizon. (Hence, a knowledge state can be inconsistent without the person being able to find this out at the current point in time.)

If the obtained knowledge state does not give contradictions *within the horizon*, then Γ_1 is accepted as the new context.

Revising the knowledge state There is, however, also the possibility that the person has found an inconsistency, i.e. he has constructed in his newly expanded knowledge state some term M such that $\Gamma_1 \vdash M : \bot$. In that case, he can decide to reject the new information and return to the previous knowledge state. But he can also decide to revise his new knowledge state in order to restore consistency. (The person may actually be able to construct more than one inhabitant of falsum; we assume that he concentrates on one of these.) The most natural thing to do, is to find one or more statements in the context representing his knowledge state, which enabled the construction of M. These statement can be located in the 'old' context, but also in the newly added piece of context, or in both. By removing one or more of these statements from his context, consistency may be regained, since this particular proof of falsum, M, cannot be constructed any more. Below we propose a syntactical iterative procedure which restores consistency. (In general, there is more than one way to regain consistency by removing statements from the knowledge state.)

The stages and decisions we distinguished above, are not intended to capture actual cognitive processes, but merely to state as clearly as possible which aspects of belief revision we do and do not consider in our formalization. For instance, the fact that the person decides which statements to remove, means that this is not decided by the formalism, in other words, we do not postulate so-called *epistemic entrenchement*. (For a comparison with standard theories of belief revision, see section 6.)

In sections 4.1 and 7.3 we discuss the various stages of dealing with information as explained just now, in more detail. We give special attention to the representation in type theory.

4.1 Adding information

The knowledge state of a person changes as new information becomes available to him. Since knowledge states are modeled by type-theoretical contexts, this means that contexts should change accordingly. In this subsection we demonstrate that type theory has the possibility to accommodate such a change in the knowledge state, viz. the addition of new information to the knowledge state.

Adding information to a type-theoretical context amounts to adding statements to this context. This does not mean that arbitrary information may be added, addition of information is subject to syntactical restrictions. We discuss this below, distinguishing between the addition of information originating from inside and from outside the knowledge state of the person.

Adding information from inside

A person is able to reason with his knowledge. For example, let us assume that the statements $A \to B : \texttt{prop}$ and $A : \texttt{prop}$ are meaningful to the person. I.e., from his knowledge state Γ, the person can derive $\Gamma \vdash A \to B : \texttt{prop}$ and $\Gamma \vdash A : \texttt{prop}$. Moreover, let us assume that the person has justifications for both propositions, since $A \to B$ and A are inhabited (e.g. $x : A \to B$ and $y : A$ occur in the context Γ representing his knowledge state). Then the person can infer that B holds, as well, expressed by the statement $xy : B$. This is the case since we have the following instance of the application rule (cf. Figure 1):

$$\frac{\Gamma \vdash x : A \to B \quad \Gamma \vdash y : A}{\Gamma \vdash xy : B}$$

This inference allows the person to combine his justification x for $A \to B$ with his justification y for A into a complex justification xy (pronounced as 'x applied to y') for the proposition B.

Note that there are no more than a small number of typing rules, which are all like the above rule in that they enable to derive a new judgement from one or more judgements which are given or derived earlier.

The judgement $\Gamma \vdash xy : B$ resulting from the person's inference as explained above, shows that the person is able to construct a justification for B on his knowledge state Γ. However, the statement $xy : B$ is not yet part of his knowledge state. To incorporate this statement, it would simply be sufficient to append it to Γ. However, for technical reasons only statements with variables as subject are allowed in the context. In order to circumvent this (technical) problem, in Section 3.3, we expanded our notion of 'context' given in Section 2.5, by allowing also a new kind of statements, called *definitions*, in the context. A definition is a statement of the form $z := E : T$, expressing that z is a name for the term E of type T. The new name z is the *subject* of the definition $z := E : T$. Formally, z is a *variable*. (This is in contrast with the good habit of calling such a defined name a *constant*.) By means of definitions, complex justifications can be abbreviated and recorded in the context. This definition mechanism is essential in the practical use of type theory for the formalization of 'bodies of knowledge', as has been shown e.g. in the Automath project [31].

A definition $z := E : T$ may be added to a context Δ whenever z is fresh with respect to Δ and $E : T$ is derivable on Δ. In the example above, this enables the person to record the inferred $xy : B$ in his knowledge state by adding the definition $u := xy : B$, using some fresh variable u. Hence, the context Γ has evolved into the context $\Gamma, u := xy : B$, reflecting the development of the person's knowledge state brought about by his reasoning. The proposition B (and its justification), which was implicit knowledge of the person (since it occurred at the right hand side of the $\vdash$), has now become explicit knowledge.

From a purely logical point of view, it may seem that adding a derived proposition to the knowledge state (making it *explicit*) does not contribute to the person's *implicit* knowledge. However, this is not the case since we assume a bound on the depth of derivations a person can perform. Under this assumption, the implicit knowledge is limited: it consists of everything a person can derive on his context *within a certain number of derivation steps*. As soon as the explicit knowledge has grown, in general there is more that can be derived by the person in the same number of steps, so the implicit knowledge has grown as well: the person's 'deductive horizon' has broadened.

Adding information from outside

The knowledge state of a person can change by reasoning (which he does himself, from the *inside*), or by information originating from the *outside*. For the latter there are two important knowledge sources: observational and communicational.

- *Observation:* A person can *recognize an object* (visually, or by any other sensory perception) in his world as belonging to a certain *set*. For example, he sees an object which he characterizes as being a ball. But he can also *obtain evidence* for *propositions* by looking at the outside world. For example, he sees that the ball is yellow.

  In both cases, the new information can be added to the context of the person by the addition of a new statement with a fresh *atomic* subject, acting as the justification. The atomic character of this justification is caused by the impossibility to decompose the observation into smaller parts.

  The two observations in the example above could e.g. be combined into the context extension $b : \ ball \ , o : \ yellow \ b$.
- *Communication:* Another manner in which a person can change his knowledge state is by information passed to him by another person. Again, this information can involve (the existence of) objects as well as (the holding of) propositions.

  For this communication it is necessary that both persons share a language in which they communicate. We assume that each person speaking this language has a mapping between the words of the language and the subjective concepts present in his knowledge state, and vice versa. In [1] a type theoretical model of communication is developed based on this assumption. In this model, the types in a person's knowledge state are communicable via the (mappings to) the common language, but the inhabitants of these types (justifications) are not. Hence the contents of a communication take the form of a (sequence of) statement(s) of which the subjects are *atomic*, since the original justifications of the 'sender' are not communicable to the 'receiver'.

  Example: in a situation after the observation of the previous example, the utterance 'The yellow ball is hollow' can lead to the following extension of the person's context: c : *hollow* b, provided that 'hollow' is a concept known to the person, and he is able to correctly match the definite description to the objects b and o in his context.

Hence, be it either observation or communication, the information to be added to a person's context has the form of a sequence of statements with atomic subjects, hence of the form $x : T$, where x is a variable; note that definitions do not play a role if we consider adding information from the outside.

However, as we said earlier, the types of the statements in the context give rise to a notion of meaningfulness. Only types 'constructable' from the statements already present in the context of a person are meaningful to him. This restricts the addition of statements originating from the outside.

Technically, this has the following form. Let Γ be the original context of the person and assume that the sequence $x_1 : T_1, \ldots, x_n : T_n$ is the information from the outside (with fresh subjects $x_1, \ldots, x_n$). Then these statements are added one by one, thus changing the knowledge state incrementally. That is to say, for each $1 \leq i \leq n$, the statement $x_i : T_i$ may only be added if

$$\Gamma, x_1 : T_1, \ldots, x_{i-1} : T_{i-1} \vdash T_i : s$$

with $s \equiv \texttt{set}$ or $s \equiv \texttt{prop}$. In other words, a statement may only be added if its type is well-formed with respect to the current knowledge state. This shows, as we said before, that new information (a sequence of statements) can only be absorbed in a step-by-step fashion (statement by statement), where the possibility to append a new statement *depends on* the information available in the context at that stage, i.e. the original context *plus* the already appended statements.

This embodies precisely the notion of incrementality, discussed in subsection 3.1, which not only applies to the case of only one 'chunk' of information from the outside (i.e. one sequence of statements) as above, but also to subsequent additions of such chunks of information. For instance, if a person is in a dialogue with another person, each new utterance he receives will be added only if it is meaningful against the background of the utterances accepted before.

Remark 4.1
In treating observation and communication, we extended the use of type theory as it is traditionally described in the literature: one usually does not take into account that information can come from outside the context. When type theory is applied to knowledge representation, one usually models (the progress of) a solitary reasoning person, who can only extend his knowledge from the inside. However, since we adopted the same well-formedness criteria as usual to adding information from the outside, the resulting context in our extension will always be syntactically correct with respect to the original type-theoretical standards. Hence, this extension of the use of type theory does not lead to an extension of the formalism. (Even the complete process of adding information from the outside can be justified in type-theoretical sense. We will not go into that here.)

4.2 The problem of revision

As we saw in the previous section, a situation in which a person has to revise his knowledge state can be characterized as follows. The person is confronted with new information (which is meaningful to him), and decides to accept it. When it turns out that the incorporation of this new information leads to inconsistency of the resulting knowledge state, the person has to remove information from this new knowledge state to restore consistency. Below we describe how this can be done by means of type theory.

Revision from a type-theoretical perspective The need for revision can originate both from the inside and from the outside. We begin by describing the situation where new information is added from outside.

Suppose that the context Γ represents the person's current knowledge state (which is consistent within his horizon) and the sequence $x_1 : T_1, \ldots, x_n : T_n$ represents the new information from the outside resulting in the context $\Gamma_1 \equiv \Gamma, x_1 : T_1, \ldots, x_n : T_n$. The inconsistency of Γ_1 manifests itself in the existence of an inhabitant of falsity which the person can construct within his horizon: there is an M such that $\Gamma_1 \vdash M : \bot$. There may be more than one such an inhabitant, but we assume that the person has chosen one of these. (We come back to this in section 5.)

The fact that all justifications are explicitly present enables the person to identify all 'suspects': the beliefs in Γ_1 that together cause the inconsistency. Since M embodies a derivation of falsity in the sense explained earlier, we find in M the justifications of all beliefs that are part of this derivation (M contains the full developmental history of the derivation). The suspect justifications occur as free variables in M, since these free variables point exactly at the premisses of the derivation of falsity: such a premiss $x : T$ gives rise to a free x in M. This is a property of the proposition as types interpretation of type theory. Moreover, the rules of type theory ensure that all free variables of M occur as subjects in Γ_1.

Example 4.2
Let $A : \texttt{prop}$ and $B : \texttt{prop}$ be statements belonging to the knowledge state (the context) and assume that the person has proofs of A, of $A \to B$ and of $\neg B$ (abbreviating $B \to \bot$, to be read as "B implies contradiction"). This is represented in the knowledge state by statements say $x : A$, $y : A \to B$ and $z : \neg B$. The rules of Type Theory enable the derivations of $\Gamma \vdash yx : B$ and $\Gamma \vdash z(yx) : \bot$. The free variables x, y and z in the 'proof object' $z(yx)$ point precisely at the propositions A, $A \to B$ and $\neg B$, which together enable the construction of the inconsistency.

Note that, given the consistency of Γ, there have to be free variables in M which occur as subjects in the *new* information $x_1 : T_1, \ldots, x_n : T_n$. (Otherwise, $M : \bot$ could already be constructed on Γ itself; this is a consequence of the Stengthening Corollary 3.5.)

New information can also originate from the inside, when a person adds a derived consequence to his knowledge state by means of a definition. This broadens his horizon and hence contradictions which were previously out of sight can now come into view (cf. section 4.1).

Example 4.3
Suppose Γ is consistent and $\Gamma \vdash N : P$ within the horizon. The result of adding $N : P$ to Γ by means of a definition is $\Gamma' \equiv \Gamma, u := N : P$. Now it is possible that there exists an M such that $\Gamma' \vdash M : \bot$ within the new horizon. As above, this M contains inhabitants of all 'suspects' as its free variables.

This shows that there is, technically speaking, no difference between revision due to information from outside and from inside. Intuitively it may seem strange that a person can be forced to revise his knowledge state by only adding a consequence of what he already knows to his knowledge state, without any external reason. However, if we take the idea of limited deductive power seriously, this is inevitable.

Restoring consistency by removing information In the above situation, when there is an M such that $\Gamma_1 \vdash M : \bot$, the person can try to regain consistency by removing one or more of the 'suspects' from Γ_1, being some of the statements $x_i : T_i$ occurring in Γ_1 where x_i occurs free in M. As we pointed out before, we assume that *the person decides* which statements he chooses to remove. Before making this choice, the person probably reconsiders the suspects, with the help of new observations or communications with others.

However, it is generally not sufficient to simply erase the chosen suspects from the knowledge state, since there may be beliefs depending on the 'suspect' beliefs. Such a dependent belief should be removed as well, since it is no longer meaningful on the knowledge state from which the suspect(s) have been erased.

A belief can depend upon another belief in two ways:

1. A belief B may contain a free variable x which is the subject variable of a statement $x : A$ preceding $y : B$ in the context.
2. If $x : A$ precedes a definition statement $z := E : C$, both E and C may contain such a free variable x.

In these cases, $y : B$ and $z := E : C$ depend on x for their well-formedness. Hence, removal of $x : A$ from the context has consequences for these statements as well. The most natural solution is to remove them.

There is a relatively simple, syntactical procedure for removing suspect beliefs *and* the beliefs depending on them, which we describe in section 5.1 The result of this procedure is a new knowledge state, Γ_2. It is, however, not necessarily the case that this Γ_2 is consistent within the person's horizon. Although the justification M of falsity is no longer constructable on Γ_2, there may have been more than one justification for falsity on Γ_1. Some of these justifications of falsity may still be constructible on Γ_2. In that case, the person chooses one of these justifications and selects a new set of suspects on which the procedure described above is repeated. Iteration leads to a sequence of knowledge states $\Gamma_1, \ldots$ which is finite, since in every iteration step at least one of the (finite number of) justifications of falsity is removed. So there is a final knowledge state Γ_n, on which no justifications of falsity are constructable. Hence, Γ_n is consistent within the person's horizon. This Γ_n is then the resulting *revised* knowledge state.

5 Belief revision

In this section we give a formal description of the process of belief revision in type theory, as described above. First we define the syntactical procedure for removing 'suspect' beliefs and the beliefs depending on them (section 5.1) stating some properties of this removal procedure. Finally, we discuss the full revision procedure, which may involve iterative removal of suspect beliefs, and we investigate the properties of the procedure.

5.1 The removal operation

We start with a knowledge state represented by a context Γ and new information represented by the sequence $x_1 : T_1, \ldots, x_n : T_n$. We add the new knowledge to the original knowledge state, obtaining $\Gamma_1 \equiv \Gamma, x_1 : T_1, \ldots, x_n : T_n$. We assume that this 'new' context Γ_1 turns out to be inconsistent and we assume that the person has chosen one or more suspect beliefs in Γ_1 which he wants to remove. Note the assumption that the suspect beliefs can be found in the *entire* Γ_1, so also among the new information: contrary to standard accounts of belief revision we do not award a special priority to the new information (cf. section 7.3).

The removal operation that we describe below results in the transformation of Γ_1 into a new context Γ_2. However, as we discuss below, regaining consistency may involve more than one such transformation, hence in our definition we define the transformation as leading from Γ_i to Γ_{i+1}.

In order to give a general definition of removal, we write a context as if all statements in the context were definitions: $y_1 := E_1 : T_1, \ldots, y_m := E_m : T_m$, with the convention that $y_l := E_l : T_l$ must be read as $y_l : T_l$ if it is not a definition and we take $FV(E_l) = \emptyset$ in the last mentioned case. ($FV(M)$ is the set of all variables occurring free in M.)

We assume that V is the set of variables which are the subjects of suspect beliefs $y_k := E_k : T_k$ in Γ_i which the person has chosen to remove. As we explained at the end of section 4, also beliefs $y_l := E_l : T_l$ *depending* on the variables in V must be removed. Below we characterize the set $\mathtt{dep}_\Gamma(V)$ consisting of V plus all subject variables of statements depending on V.

We start with the definition of the notion 'subcontext'.

DEFINITION 5.1
Let $\Gamma \equiv \Delta_1, y := E : T, \Delta_2$ and $\Gamma' \equiv \Delta_1, \Delta_2$ or $\Gamma' \equiv \Delta_1, y : T, \Delta_2$. Then $\Gamma' \subset \Gamma$. The relation $\subseteq$ is the reflexive and transitive closure of $\subset$. If $\Gamma_1 \subseteq \Gamma_2$ we say that Γ_1 is a *subcontext* of Γ_2.

Next we define the dependency relation $\leq$, a partial order between subject variables of a context Γ.

DEFINITION 5.2
Let $\Gamma \equiv \Delta_1, y := E : T, \Delta_2$. Then $\mathtt{def}_\Gamma(y) = E$, $\mathtt{type}_\Gamma(y) = T$ and $\mathtt{stat}_\Gamma(y) = (y := E : T)$. For y and $z \in \mathtt{dom}(\Gamma)$ we say that $y < z$ if $y \in \mathtt{FV}(\mathtt{def}_\Gamma(z) \cup \mathtt{type}_\Gamma(z))$.(For convenience, we write '$<$' instead of $<_\Gamma$.) The relation $\leq$ is the reflexive and transitive closure of $<$. The set $\mathtt{dep}_\Gamma(y)$ is $\{z \in \mathtt{dom}(\Gamma) | y \leq z\}$. Moreover, $\mathtt{dep}_\Gamma(V)$ is $\bigcup_{y \in V} \mathtt{dep}_\Gamma(y)$, for $V \subseteq \mathtt{dom}(\Gamma)$.

Note that the set of variables depending on a set of variables V, *includes* V itself.

Next, we define a deletion operator $\mathtt{del}$, erasing statements from a context, and the removal operator '$\backslash$'.

DEFINITION 5.3
For domain variable y of $\Gamma \equiv \Delta_1, y := E : T, \Delta_2$, we define $\Gamma - \mathtt{stat}_\Gamma(y)$ as Δ_1, Δ_2. For a set W of domain variables of Γ, we define $\mathtt{del}_\Gamma(W)$ as $\Gamma - \bigcup_{y \in W} \mathtt{stat}_\Gamma(y)$. For a context Γ and a set $V \subseteq \mathtt{dom}(\Gamma)$, the *removal operation* ' $\backslash$ ' is defined by $\Gamma \backslash V = \mathtt{del}_\Gamma(\mathtt{dep}_\Gamma(V))$.

So, $\Gamma \backslash V$ is the context resulting from removing all statements depending on the set V of chosen subject variables, from Γ.

As explained in section 4.1, knowledge states are incremental, in the sense that the type of each statement should be meaningful given the statements preceding it. In type theory this is expressed by *legality* given in Definition 2.23 and which satisfies the important Context Lemmma 3.6. The removal operator applied to a legal context, results in a new, legal subcontext:

LEMMA 5.4
Let Γ be a context and $V \subseteq \mathtt{dom}(\Gamma)$. Then $\Gamma\backslash V \subseteq \Gamma$. Moreover, if Γ is legal, then $\Gamma\backslash V$ is legal.

PROOF. For the second part: Subsequently delete all $\mathtt{stat}(y)$ for $y \in \mathtt{dep}_\Gamma(V)$ from Γ, from right to left, using Strengthening Corollary 3.5. ∎

The removal operator has the nice property that the result of subsequent applications to V and W is the same as applying it in the reverse order, or by applying it to the union of V and W:

LEMMA 5.5
If Γ is legal and V and W are subsets of $\mathtt{dom}(\Gamma)$, then $(\Gamma\backslash V)\backslash W = (\Gamma\backslash W)\backslash V = \Gamma\backslash(V \cup W)$.

PROOF. By the definition of $\backslash$ and basic set theory. ∎

5.2 The revision procedure

In this section we show how the removal operator can be used to regain consistency. We assume that a person has originally a legal and *consistent* knowledge state Γ. He extends his context Γ with new information $x_1 : T_1, \ldots, x_n : T_n$, obtaining $\Gamma_1 \equiv \Gamma, x_1 : T_1, \ldots, x_n : T_n$. Let's assume that Γ_1 is legal again, but that it has become inconsistent: he can now construct an M such that $\Gamma_1 \vdash M : \bot$. (Note: in this subsection we forget about the 'horizon' of a person, i.e. the limited deduction power of a human being; we consider this horizon in the following subsection.) We consider two cases in both of which the proof M of falsity is no longer derivable on the resulting context Γ_2:

- The person chooses to remove a *single* subject variable z occurring freely in M, plus all statements depending on this z. Hence, he obtains $\Gamma_2 \equiv \Gamma_1\backslash\{z\}$ as his new context. Note that the chosen variable z may be the inhabitant of a statement in the original context Γ or of a statement $x_i := E_i : T_i$ which is part of the extension. In the latter case, $\mathtt{dep}_{\Gamma_1}(z)$ contains only variables occurring as subjects in the extension. In the former case, however, $\mathtt{dep}_{\Gamma_1}(z)$ may contain subject variables of Γ *as well as* subject variables of the extension. Hence, the removal operation may change the new information in *both* cases.
- The person chooses a non-empty *set* V of variables occurring freely in M and obtains $\Gamma_2 \equiv \Gamma_1\backslash V$ as his new context. Note that by lemma 5.5, the removal of V has the same effect as removing the separate elements of V, one by one, in *any* order. (This also holds if V is the set of *all* free variables in M.)

The above does not guarantee that Γ_2 is consistent: it may be the case that the person can still construct a proof of falsity, say M', on Γ_2. Then the person can repeat the removal operation with one or more free variables occurring in M', and so on obtaining a sequence of contexts $\Gamma_1, \Gamma_2, \ldots$, where each Γ_{i+1} is a legal subcontext of Γ being properly 'smaller' (i.e. contains fewer statements) than Γ_i. It follows that the sequence $\Gamma_1, \Gamma_2, \ldots$ is finite, so that a context Γ_n which is consistent is finally obtained. (In the extreme case $\Gamma_n = \varepsilon$, but there is no proof of falsity on the empty context ε by the consistency of type theory.) This implies:

LEMMA 5.6
Iterated application of the removal operation terminates resulting in a consistent knowledge state.

In other words, it is a *revision procedure*. It is interesting to note that this iteration can be summarized in a single application of the removal operation: Let's call the non-empty set of variables that the person chooses to remove in the transition from Γ_i to Γ_{i+1}, V_i (which can be a singleton set). Then $\Gamma_{i+1} = \Gamma_i \backslash V_i$. However:

LEMMA 5.7
Successively removing V_i from Γ_i for $i = 1, \ldots, n-1$, leads to the same result as removing the union of all V_is from Γ_1: I.e. $\Gamma_n = \Gamma_1 \backslash \bigcup_{i=1}^{n-1} V_i$.

PROOF. This is again a consequence of lemma 2. ■

In this section we assumed that it is the *person* who makes the decision about which statements to remove, and not the formalism. We gave arguments for this point of view in section 4. However, in comparing our system with others in the literature we will in section 7.4 discuss formal *heuristics* for making these decisions.

5.3 Revision with horizon

In the previous subsection we assumed that the person is 'omniscient' in the sense that he is able to provide a proof of falsity at any time, if there *exists* one. This, of course, is not realistic. For this reason we introduced in the beginning of section 4 the notion of 'horizon' for the person. If we look at the revision procedure, the presence of a horizon has important consequences.

Firstly, a knowledge state Γ has only a limited number of consequences within a given horizon. We formulate this as a theorem, provable by combinatorial arguments:

THEOREM 5.8
Given a context Γ and a number h limiting the derivation depth of derivations on Γ ('the distance to the horizon'), there is a *finite* number of statements derivable on Γ (modulo α-conversion).

Note that we do *not* consider the full deductive closure of Γ, which possibly corresponds with an 'infinite horizon', which is no horizon at all. For convenience, we denote the finite set of derivable statements from context Γ (the set of consequences of Γ) within horizon distance h by $Conseq_h(\Gamma)$.

COROLLARY 5.9
Given a context Γ that is inconsistent within horizon distance h, there is a finite number of inhabitants of falsity ('proofs of falsity') (modulo α-conversion). I.e., there are finitely many terms M such that $M : \bot \in Conseq_h(\Gamma)$.

By application of the revision procedure, statements are removed from the context Γ. This will eliminate a (number of) proof(s) of falsity, but the question arises whether there are *new* proofs of falsity on the revised (smaller) context. This is not the case:

Theorem 5.10
If $\Gamma\backslash V$ is the result of revising Γ with respect to V, then there is no statement derivable within horizon distance h on $\Gamma\backslash V$ which was not already derivable within horizon distance h on Γ. I.e., $Conseq_h(\Gamma\backslash V) \subseteq Conseq_h(\Gamma)$.

Proof. *Note that* $\Gamma\backslash V \subseteq \Gamma$ *by lemma 5.4. For any two PTS-contexts* Δ *and* Δ' *the so-called Thinning Lemma holds: if* $\Delta' \subseteq \Delta$ *and* $\Delta' \vdash A : B$*, then* $\Delta \vdash A : B$*. Hence if* $\Gamma\backslash V \vdash A : B$ *then* $\Gamma \vdash A : B$*. However, if we regard the horizon distance, it might still be possible that there exists a statement* $A : B$ *which is derivable on* $\Gamma\backslash V$ *in at most h steps, and on* Γ *in more than h steps (due to extra steps needed to 'retrieve' the premisses on the larger context). We assume, however, that the axiomatization of Type Theory is such that the Start-rule allows any number of Weakenings. In that case, a derivation of* $\Gamma\backslash V \vdash A : B$ *can always be 'copied' into a derivation of* $\Gamma \vdash A : B$ *with the same number of derivation steps.* ∎

Corollary 5.11
The removal procedure does not allow the introduction of new proofs of falsity.

Corollaries 5.9 and 5.11 imply the following theorem, which says that we can always reach a consistent context *in one revision step*:

Theorem 5.12
Given an inconsistent context Γ and a horizon distance h, there exists a set of variables V such that $\Gamma\backslash V$ is consistent within the same horizon distance.

Proof. *Take* V *to be the set of all free variables occurring in all proofs of falsity which can be derived on* Γ *within horizon distance h. By Corollary 5.9, this set is finite and by the definition of the revision procedure, none of these proofs of falsity are constructable on* $\Gamma\backslash V$*. By Corollary 5.11, there are no new proofs of falsity on* $\Gamma\backslash V$*, hence* $\Gamma\backslash V$ *is consistent within horizon distance h.* ∎

6 Situating our approach

In this paper, we presented an approach to belief revision based on type theory. As far as we know, this approach is new. In the setting of type theory, justifications of beliefs are 'first class citizens', which is not the case in current approaches to belief revisions. In this section we discuss the relations between our approach and well-known approaches from the literature. We take [18] as our guideline.

6.1 Belief bases with justifications

By the methodological taxonomy of [18], our approach has these characteristics:

- Beliefs are represented as statements in type theory, a person's belief state as a type-theoretical context (section 4). The result of a belief change operation is again a type-theoretical context (section 5.2).

- The statements that are elements of the context representing a person's belief state, represent his *explicit* beliefs. Beliefs derivable from these statements are his implicit beliefs (section 3.2). Contrary to standard practice, we assume the deductive powers of the person are limited by a deductive horizon and only statements that are derivable within this horizon count as his implicit beliefs.
- Our theory does not prescribe how choices are made concerning what beliefs to retract. It gives a set of candidates for retraction, but leaves the actual choice to the person (Section 5.2). One can give heuristics for this choice (Section 7.4).

Gärdenfors and Rott mention four *integrity constraints* guiding the construction of belief revision formalism:

- *The beliefs in the data base should be kept consistent whenever possible.* We adhere to this constraint taking 'consistent' to mean: 'consistent with respect to the limited deductive powers of the person'.
- *If the beliefs in the data base logically entail a sentence, then this sentence should be included in the data base ('deductive closure').* It will be clear from our earlier comments (sections 4 and 5.3) that we do not subscribe to this point of view. However, it is possible to explicitly include a derived belief (to be precise: derived within the person's horizon) in the knowledge state by means of a definition (section 4.1).
- *The amount of information lost in a belief change should be kept minimal.* In accordance with the fact that our theory says nothing about extra-logical factors governing the choice of beliefs-to-be-retracted, there is no notion of minimality inherent in our theory.
- *In so far as some beliefs are considered more important or entrenched than others one should retract the least important ones.* In line with our previous comment, a notion of extra-logical preference like entrenchment should in our opinion not be part of a theory as it belongs to the realm of heuristics.

The choices we made above imply that we work with the so-called *belief bases*: the knowledge state of a person is represented by a finite set of sentences, a context Γ. The belief set of the person consists of his explicit beliefs (statements in Γ) and his implicit beliefs (statements derivable on Γ within the horizon, i.e. $Conseq_h(\Gamma)$). Note that $\Gamma \subseteq Conseq_h(\Gamma)$: every explicit belief in the context Γ is derivable on Γ, and is hence *also* implicit. Therefore we can represent a person's belief set by $Conseq_h(\Gamma)$.

Since we choose to represent justifications for beliefs explicitly, as inhabitants, in the knowledge state, our approach is closely related to what is called *Foundations Theory* in the literature, see e.g. [17].

6.2 The relation with Foundations Theory

Foundations Theory is based on the principle that belief revision should consist in giving up all beliefs that no longer have a satisfactory justification, and in adding new beliefs that have become justified. This principle has a number of consequences:

- *Disbelief propagation* If in revising a knowledge state a certain belief is retracted, not only this belief should be given up, but also all beliefs depending on this belief for their justification. Since our theory has an explicit representation of justifications, this propagation can be captured syntactically, as was shown in definition 5.2, by means of the relation $\leq$. Hence, our approach does not have the drawbacks that are often associated with disbelief propagation, viz. 'chain reactions' and 'severe bookkeeping problems'.

- *Non-circularity.* Since beliefs can depend on other beliefs for their justification, we should be careful that the dependency graph is well-founded, i.e. does not contain circularities. In our approach such circularities cannot occur, since they are ruled out by the well-formedness requirements for the type-theoretical contexts (section 4).

- *Multiple justifications.* A belief may be supported by several independent beliefs. The removal of one of those justifications does not automatically lead to giving up the belief. This characteristic is reflected in our approach, where a belief may have more than one inhabitant. Suppose that a person has two justifications for the belief that A holds on his knowledge state Γ, for example: $\Gamma \vdash M : A$ and $\Gamma \vdash N : A$. Since the free variable sets of M and N may be disjoint, it may be possible to retract the justification M of A, while retaining N and hence the belief that A (see section 5.2).

There is a well-known problem in Foundations Theory, following from the hypothesis that all beliefs must have a justification. This induces a distinction between beliefs: some beliefs are justified by one or more other beliefs, but there must also exist beliefs which are justified 'by themselves'. These so-called *foundational beliefs* are considered to be 'self-evident', they need no further justification.

In Foundations Theory, justification is a relation on the level of the beliefs. In type theory, however, justifications are explicitly represented by terms *inhabiting* the beliefs they justify. The distinction between foundational and other beliefs is reflected in type theory in the structure of the term inhabiting the belief:

- *Atomic justifications.* If the term inhabiting the belief is a constant or a variable, the justification cannot be further analyzed. This corresponds to the foundational beliefs, but only to a certain extent: it does not imply that these beliefs are necessarily self-evident. The atomic justification simply reflects the person's decision to adopt the belief in its own right, e.g. on the basis of an observation, communication or an act of will. (See also section 2.)

- *Composite justifications.* If the term inhabiting the belief is a composite term, the justification can be analyzed according to the structure of the term. These terms occur in the context in *definitions*, e.g. in the statement $y := E : T$, where E is a composite justification for T. One can find the inhabitants of the other beliefs supporting the belief that T, as the free variables occurring in E.

Thus the justification relation from Foundations Theory becomes a relation between inhabitants of beliefs in type theory. This relation is captured by the dependency relation $\leq$ of definition 5.2.

7 Comparing operations for belief change

Before we can compare the formal properties of our revision procedure with those of the literature, we must formulate our equivalents of the three standard belief change operations: expansion, contraction and revision.

- *Expansion: Adding a new sentence A to the belief base K, regardless of the consistency of the resulting belief base.* The result is usually denoted by $K + A$.

  In our type-theoretical setting, expansion is just addition of either a statement or a definition to the context: Γ changes into $\Gamma, x : A$ (with x fresh), or into $\Gamma, x := M : A$. In the first case new information originating from outside is added, in the second case a consequence of the belief base is made explicit by adding it to the context.

  Note that, in both cases, the type A must already be well-formed with respect to Γ, i.e. $\Gamma \vdash A : s$ with s a sort in the set of sorts $\mathcal{S}$ of the type system (cf. sections 3.2 and 4.1). In the second case, $x := M : A$ may only be added when $\Gamma \vdash M : A$ is derivable. This again gives a well-formedness guarantee.

  NOTATION: The type-theoretical analogue of Expansion will be denoted by $Exp_{x:=M:A}(\Gamma; \Gamma')$ if the expansion of Γ with the statement or definition $x := M : A$ yields Γ'. Hence, $\Gamma' \equiv \Gamma, x := M : A$.

- *Contraction: Retracting some sentence A from the belief base K, as well as sentences depending on A (without adding new beliefs).* This is denoted by $K \dot{-} A$.

  In type theory, retracting has to be done with statements instead of formulas. Moreover, given a context Γ and a horizon depth h, there can be several terms inhabiting a belief A that is to be retracted. There is a *set* of terms t such that $t : A \in Conseq_h(\Gamma)$. If we take retraction to mean that *no* statement $M : A$ should be derivable any more, we need a retraction procedure similar to the one described in section 5.2. That is, the person iteratively chooses variables occurring free in such terms t inhabiting A and removes them from Γ, in order to eliminate evidence for A.

  Formally, we can say that there is a set $V_A := FV\{t | t : A \in Conseq_h(\Gamma)\}$. The variables chosen by the person together constitute a subset V of V_A (cf. Lemma 4). Retraction of A with respect to Γ then amounts to a removal $\Gamma \backslash V$ with V chosen such that $\neg\exists_t (t : A \in Conseq_h(\Gamma \backslash V))$.

  Note: In its generality, this procedure always gives the desired result. There is, however, a slight complication: there are sentences which we never want to be contracted, for example tautologies. How we can prevent in type theory that this kind of sentences can be retracted, is discussed in section 7.2.

  NOTATION: The type-theoretical analogue of Contraction is denoted by $Ctr_A(\Gamma; \Gamma')$, if Γ' is the result of contracting Γ with respect to A. In case $A \notin Conseq_h(\Gamma)$, we take Γ' to be Γ.

- *Revision: Adding a new sentence A to the belief base K while maintaining consistency, by (possibly) deleting a number of sentences in K.* This is denoted by $K * A$.

  In the standard account, revision is related to contraction and expansion by means of the Levi-identity: $K * A = (K \dot{-} \neg A) + A$. This implies, that for belief bases, revision can be defined as a two step procedure:

1. *Contract by* $\neg A$ 2. *Expand by* A

We can match this so-called internal revision [21] via the two type-theoretical operations defined above:

1. $Ctr_{\neg A}(\Gamma; \Gamma')$ 2. $Exp_{x:=M:A}(\Gamma'; \Gamma'')$

Note that this procedure will always lead to a context (Γ'') containing the new information ($x := M : A$), whereas the procedure described in sections 5.2 and 5.3 did not, since there it was possible that (parts of) the new information were removed as well, if this information contributed to the inconsistency. In literature, this alternative approach is known as 'semi-revision'. In section 7.3 we will show that the type-theoretical version of revision developed in this paper closely resembles the semi-revision operation *consolidation* of [21]. Anticipating on this, we introduce the following.

NOTATION: The type-theoretical analogue of Revision (i.e., Contraction by $\neg A$ and Expansion by A) is denoted by $Rev_{x:=M:A}(\Gamma; \Gamma')$, if Γ' is the result of revising Γ with respect to $x := M : A$.

Finally we note that the operations of expansion and contraction, and hence revision, described above can also be executed with new information consisting of a *sequence* of statements ($x_1 := M_1 : A_1, \ldots, x_i := M_i : A_i$), rather than a single statement ($x := M : A$). From a type-theoretical point of view, this is a natural generalization. Moreover, experiences obtained in formalizing the addition of outside-information (as described in section 4.1) to type-theoretical knowledge states, suggests that such information generally takes the form of a sequence of statements.

Now we have given our equivalents of the standard belief change operations, expansion, contraction and revision, we give a more detailed comparison between the two approaches in order to position our approach with respect to the literature. We concentrate on the results of Gärdenfors [18] and Hansson [21].

7.1 Expansion

In the standard approach, expansion is the set-theoretical addition of a sentence to set of propostions representing a person's belief base. In the type theoretical approach it is the addition of a statement to the context representing a person's belief base. As explained above, the type theoretical addition requires that the new statement is well-formed with respect to the existing context, which ensures that the added information is meaningfull to the person. Assuming that this the case, as is usually done, expansion behaves the same in both approaches.

7.2 Contraction

We now look at the rationality postulates for contraction as they are reformulated for belief bases in [18]. As already remarked earlier, our approach is more fine-grained than that of Gärdenfors, because we deal with specific proofs of propositions, whereas the standard approach does only considers (sets of) propositions. Hence, when Gärdenfors contracts with respect to a proposition A, from our perspective, he implicitly quantifies over *all* proofs of A. This difference also plays a role in the formulation of the postulates themselves.

In some of the Gärdenfors postulates, conditions occur of the form $\vdash A$ and $\not\vdash A$. Type-theoretically, we take these to state that there *exists* respectively *doesn't exist* a proof object for the type A within the horizon. Moreover, the fact that A is or isn't a tautology, suggests that this proof object can (or cannot) be constructed on the empty context ε. However, in type theory the type A itself must be well-defined before we can think about the construction of inhabitants of A. Hence, we need some *initial context* Γ_{init} which ensures the well-definedness of all propositions: $\vdash A$ is translated into $\exists_M(\Gamma_{init} \vdash M : A)$ and $\not\vdash A$ into $\neg\exists_M(\Gamma_{init} \vdash M : A)$.

Of course, statements in the initial context should not be contracted in a revision process, since this initial context acts as a kind of 'axiom base' for the well-definedness of the propositions. The above contraction procedure $Ctr_A(\Gamma;\Gamma')$, will not consider variables inside Γ_{init}, since the statements of Γ_{init} are at the wrong level of typing to have their subjects appear in terms inhabiting propositions (cf. section 2).

Note that if A is a tautology, there exists a proof object in which no free variables occur: $\exists_M(\Gamma_{init} \vdash M : A)$ where $V_A = \emptyset$. Since M cannot be blocked by removing variables in V_A from the context, we cannot contract over tautologies. On the one hand this is a good thing: one does not want to lose tautologies. On the other hand, this has as a consequence that Contraction becomes a *partial* operation, which may be unsuccessful!

Below we present the Gärdenfors postulates for belief bases as given in [18], followed by their type-theoretical translation and a discussion of their validity. The original postulates quantify over all sentences A and belief sets H, their translations over all types A and contexts Γ (where $\Gamma \supseteq \Gamma_{init}$). In addition, the postulates are stated using $Cn(H)$, the deductively closed set of consequences of H (i.e. with infinite horizon). In the translation of e.g. Gärdenfors's ($H\dot{-}3$)-postulate we take $A \notin Cn(H)$ to mean that there exists no proof object of type A (within the horizon) on the person's context, $\neg\exists_N(N : A \in Conseq_h(\Gamma))$.

($H\dot{-}1$) $H\dot{-}A$ is a belief set.

Its translation is:

If $Ctr_A(\Gamma;\Gamma')$, then Γ' is a well-formed context.

This holds: Assume $Ctr_A(\Gamma;\Gamma')$, then there exists some set $V \subseteq V_A$, possibly empty, such that $\Gamma\backslash V \equiv \Gamma'$. By Lemma 1, Γ' is a well-formed context.

($H\dot{-}2$) $H\dot{-}A \subseteq H$.

Its translation is:

If $Ctr_A(\Gamma;\Gamma')$, then $\Gamma' \subseteq \Gamma$.

This follows from the definition of the removal-operation (definition 5.2).

($H\dot{-}3$) If $A \notin Cn(H)$, then $H\dot{-}A = H$.

Its translation is:

If $\neg\exists_N(N : A \in Conseq_h(\Gamma))$ and $Ctr_A(\Gamma;\Gamma')$, then $\Gamma \equiv \Gamma'$.

This holds: Assume $\neg\exists_N(N : A \in Conseq_h(\Gamma))$ and $Ctr_A(\Gamma;\Gamma')$ and suppose $\Gamma \not\equiv \Gamma'$. Then (see $H\dot{-}2$) Γ' is a proper subcontext of Γ. Hence there is some variable z occurring in Γ as a subject, such that $z \in V$, where V is the set of variables chosen to be removed and $z \in V$ not in Γ'. Hence z must have occurred free in some term N such that $\Gamma \vdash N : A$ within the horizon, but then $\exists_N(N : A \in Conseq_h(\Gamma))$. Contradiction.

($H\stackrel{\circ}{-}4$) If $\not\vdash A$, then $A \notin Cn(H\stackrel{\circ}{-}A)$.

Its translation is:

If $Ctr_A(\Gamma;\Gamma')$, then $\neg\exists_M(M : A \in Conseq_h(\Gamma'))$.

This postulate holds by our definition of contraction.

Note that the condition $\not\vdash A$ ('A' is not a tautology) is implicitly present in our translation, because this is implied by the condition $Ctr_A(\Gamma;\Gamma')$. In fact, if A *is* a tautology, then A has a proof object, but this proof object has no free variables. Therefore the set V_A is empty and hence Contraction of A as described before is not possible (there is no Γ' such that $Ctr_A(\Gamma;\Gamma')$).

($H\stackrel{\circ}{-}5$) $H \subseteq (H\stackrel{\circ}{-}A) + A$.

Its translation is:

If $Ctr_A(\Gamma;\Gamma')$, then $\Gamma \subseteq \Gamma', z : A$.

Note that we have to add a proof object z for A. We could not use a definition $z := M : A$, since this implies that $\Gamma' \vdash M : A$ for some M, which contradicts $Ctr_A(\Gamma;\Gamma')$.

This postulate, which has a controversial status in the literature (in fact: base contractions generally violate it), does not hold here. A simple counterexample is the following: Take $\Gamma \equiv \Gamma_{init}, x : B \to A, y : B \vdash xy : A$, then $Ctr_A(\Gamma,\Gamma')$, where $\Gamma' \equiv \Gamma_{init}$, but $\Gamma \not\subseteq \Gamma_{init}, z : A$.

($H\stackrel{\circ}{-}6$) If $\vdash A \Leftrightarrow B$, then $H\stackrel{\circ}{-}A = H\stackrel{\circ}{-}B$.

Its translation is:

If $\exists_N(\Gamma_{init} \vdash N : A \Leftrightarrow B)$ and $Ctr_A(\Gamma;\Gamma')$ and $Ctr_B(\Gamma;\Gamma'')$, then $\Gamma' \equiv \Gamma''$.

This postulate does not hold in general, but there is a case in which it holds, as we explain below.

First, observe that in type theory we have to do work to transform proofs of A into proofs of B (and vice versa) by means of the proof N of the equivalence of A and B which contains subproofs N_1 for $A \to B$ and N_2 for $B \to A$. Then for example: If $\Gamma \vdash M : A$ for some M, then $\Gamma \vdash N_1 M : B$ (and vice versa).

We call M a *direct* proof of A and $N_1 M$ an *indirect* proof of B. Note that transforming a direct proof of A into an indirect proof of B involves one extra proof step. Hence, this can lead to a situation in which the direct proof is within the horizon, whereas the indirect proof is not.

Disregarding this horizon problem, the postulate still does not hold in general: in order to block all proofs of B, all proofs of A also have to be blocked. Hence, a set V will have to be chosen which is a subset of the union of the variables occurring free in all proofs of A *and* all proofs of B, i.e., $V \subseteq (V_A \cup V_B)$. However, it might still be possible to find different subsets V_1 and V_2 which both block all proofs of A and B.

Example: $\Gamma \equiv \Gamma_{init}, x : C \to A, y : C, z : D \to B, u : D$, and $\Gamma \vdash N : A \Leftrightarrow B$. Then $V_A = V_B = \{x, y, z, u\}$. Now take $V_1 = \{x, z\}$ and $V_2 = \{y, u\}$. It is easy to check that both V_1 and V_2 block all proofs of A and B. If we take $\Gamma' \equiv \Gamma \backslash V_1$ and $\Gamma'' \equiv \Gamma \backslash V_2$, then $Ctr_A(\Gamma;\Gamma')$ and $Ctr_B(\Gamma;\Gamma'')$, but $\Gamma' \not\equiv \Gamma''$.

However, the postulate *does* hold if we use the 'safe contraction' described in section 7.4, i.e. take $V_1 = V_2 = V_A = V_B$, then $\Gamma' \equiv \Gamma''$.

Here we end our discussion of the basic postulates $H\overset{\circ}{-}1$ to $H\overset{\circ}{-}6$ for base contraction. There exist two more (non-basic) postulates, $H\overset{\circ}{-}7$ and $H\overset{\circ}{-}8$, concerning conjunctive formulas $A \wedge B$. We do not discuss those here for two reasons: as remarked above, the type-theoretical notion of contraction can easily be generalized to a sequence of statements, so that there is no need to give a special status to the $\wedge$-connective; moreover, it would require us to go into the technical details of coding conjunction in type theory, which does not serve the purpose of this paper.

Concluding, as in most approaches to base revision in the literature, postulates $H\overset{\circ}{-}1$ through $H\overset{\circ}{-}4$ are satisfied in the type-theoretical translation, but $H\overset{\circ}{-}5$ does not hold. In addition, the type-theoretical equivalent of 'safe contraction' satisfies $H\overset{\circ}{-}6$. This exactly reflects Theorem 5.4.1 of [18].

7.3 Revision

In the standard account of revising a belief base K with new information A, the new information is always accepted and beliefs in K are abandoned to maintain consistency. Objections have been raised to this account, on the grounds that too much priority is given to new information [21]: at each stage, new information is completely trusted. However, this complete trust is only temporary: once the new information is incorporated in the belief base, it is itself susceptible to abandonment when in the next stage even newer information becomes available. This seems awkward.

We agree with these objections. Moreover, this emphasis on 'new information' has a number of additional undesired consequences from our point of view. Firstly, the new information always has to be accepted as a whole, whereas in our approach it is a possible outcome of revision that the person accepts only part of the new information. The standard account is also too absolute in another respect: because of the unlimited deductive power assumed in this approach, the person can detect beforehand whether a piece of new information is inconsistent with his current belief base, and hence whether revision should be carried out. Under the more realistic assumption of the deductive horizon, it is not possible to do this consistency check once and for all: inconsistencies, and hence the need for revision, may arise as proofs of falsity turn up inside the horizon. Finally, thinking of standard belief revision in the setting of communication, a person would be forced to accept every utterance by his dialogue partner(s), even if accommodating this information requires a major reconstruction of his own belief base. Therefore, new information and information in the belief base should be treated equally by the revision operation.

Revision procedures which do not necessarily accept the new information are known in literature as *non-prioritized* revision procedures. Hansson was one of the first to consider this kind of belief revision [20], and in recent years a number of different non-prioritized approaches have been developed, see [22]. For belief bases, a non-prioritized form of revision called *semi-revision* can be specified as a two-stage procedure [21]:

1. *Expand by A*
2. *Make the belief base consistent by deleting either A or some original belief(s)*

Compared to the revision procedure formulated at the beginning of section 7, the order of the steps is reversed[4] and the second step has been modified. The operation

[4]Reversing the order alone yields *external revision*, [21]

performed in the second stage is called *consolidation*, [21], and can be carried out by contracting over falsehood. In our approach, the procedure looks like:

1. $Exp_{x:=M:A}(\Gamma;\Gamma')$ 2. $Ctr_{\perp}(\Gamma';\Gamma'')$

In other words, revision and contraction are related by the following identity:

$$Rev_{x:=M:A}(\Gamma;\Gamma') = Ctr_{\perp}(\Gamma, x := M : A;\Gamma')$$

This is exactly the revision procedure described earlier in sections 5.2 and 5.3. First the new information, one or more statements, is added to the context Γ, then a number of statements from the expanded context is removed to block the construction of inhabitants of falsity.

There is a close resemblance between our revision procedure and that of [21], called *kernel consolidation*. This correspondence is given in Appendix A of this paper.

7.4 Heuristics

What we have done so far does not add up to a theory of belief revision in the traditional sense. We have shown how a person can find the suspect beliefs when his belief state has become inconsistent, and how he can remove a number of the suspects to regain consistency, but our revision procedure does not tell the person *which* suspects to remove. Standard approaches have a parametric selection mechanism which embodies some notion of “rational choice” between the various possibilities for revision in any given situation. Given a value for their parameters they select one “optimal” revision outcome. They usually introduce extra-logical structure in the belief state, and are computationally unwieldy. The underlying view is that of a solitary reasoner who has to solve the inconsistency in splendid isolation, using his infinite reasoning powers and looking only at the beliefs in his (infinite) belief state. Only recently, papers have started to appear that question some of these idealizations, and in which belief change operations are defined for resource-bounded agents, see e.g. [10]. Our concern is with agents who have finite belief states (including justifications), finite computational resources, and who have access to the world by means of observation and communication. Such agents have possibilities to (re)evaluate the various suspects, by performing observations/tests or by communicating with other agents, and a theory of belief revision cannot and should not prescribe how they make their choices. Strategies used by an agent to make these choices are not part of the theory, if they can be captured formally they could be used as heuristics on top of the theory. In this section, we briefly discuss how some selection mechanisms from standard approaches mentioned in [18] fit into our account as heuristic principles.

In so-called *(partial) meet contraction*, the idea is that the optimal contraction or revision is the one that requires the smallest number of insertions and/or deletions in the belief state. These contraction operations were originally defined for deductively closed belief sets, rather than belief bases. They start from the maximal subsets of a belief set that do not imply the proposition that is to be removed. In general, there can be quite a few of these. Picking an arbitrary maximal non-implying set as the result of the operation (choice contraction), will often yield a new belief set that is too large. In meet contraction, an intersection of maximal non-implying sets is taken, to obtain a new belief set based on the beliefs the non-implying sets have in common.

Taking the intersection of *all* maximal non-implying sets (full meet contraction) can result in an empty set. Alchourrón, Gärdenfors, and Makinson introduced partial meet contraction [2] in which a selection function picks out a class of "best" or "most interesting" maximal non-implying subsets. These selected sets are then intersected to obtain the new belief set. (For a fresh look at (full) meet contraction for belief bases, see [14]).

The minimality criterion can be applied in the type theoretical approach. Given one particular proof of inconsistency, $\Gamma \vdash M : \bot$, removing any one of the statements of which the subjects occur free in M is sufficient to block this particular proof. However, these statements may have different numbers of statements depending on them in Γ, and so one could prefer to remove the statement with the least number of dependents to minimise the deletions from the belief state. In cases where more than one proof of falsity has to be blocked, a "blocking" subset has to be chosen from the set of all variables occurring free in these proofs. When there is more than one subset that does the job, one could again prefer the subset with the smallest number of statements (possibly taking the number of dependent statements into account).

As in the standard approach, this criterion will not always yield a single optimal solution. It is possible to end up with two or more minimal sets of statements whose removal will restore consistency. To overcome this indeterminism, additional structure is introduced in the belief state. The central idea in this construction is known as *epistemic entrenchement*: "not all sentences that are believed to be true are equal value for planning of problem-solving purposes, but certain pieces of knowledge and beliefs about world are more important than others when planning future actions conducting scientific investigations or reasoning in general" [17]. In performing contraction or revision, the beliefs that are given up should be the ones with the lowest degree of epistemic entrenchement. Although in our opinion such an ordering of epistemic entrenchement of the beliefs in the belief state cannot be given once and forall independent of the current goals and activities of the agent performing the contraction or revision, such an ordering could in principle be added to the context representing the agent's belief state. Note that the imposed entrenchment ordering has to respect the dependency relations between the beliefs in the context: if a belief $y := N : B$ depends on a belief $x := M : A$, then $y := N : B$ should not be epistemically more entrenched than $x := M : A$ since removing $x := M : A$ without removing $y := N : B$ will result in a context which is not well-formed.

Another idea that can be applied, at least in spirit, in the type theoretical setting is that of *safe contraction*: a proposition B is safe with respect to a proposition A if it cannot be blamed for the derivability of A. To contract over A, all propositions that are not safe with respect to A have to be removed. This approach, introduced by Alchourrón and Makinson [3], starts from the so-called "entailment sets": minimal subsets of the belief state that entail the proposition to be removed. An element B of the belief state is said to be safe with respect to the proposition A if B is not a minimal element of any entailment set of A. The minimality is determined with respect to an acyclic ordering of the beliefs in the belief state, expressed by means of a relation '$<$'. This ordering can be seen as a form of the epistemic entrenchment described above, with $A < B$ meaning that A is "less secure, plausible or reliable" than B.

There is an obvious way to translate this idea to our approach to revision: a belief

$x := M : A$ is safe if it cannot be blamed for the fact that a proof object for $\perp$ can be constructed on the belief state Γ. The simplest interpretation of "being to blame" for a statement in context would be "to have its subject appear as a free variable in a proof object for $\perp$". Hence the simplest form of safe contraction would be to remove all statements of which the subjects appear free in a proof object for $\perp$ and their dependents from the context. However, this does not suffice if all statements that are removed themselves depend upon earlier statements in context, since the proof object for $\perp$ could be rebuilt from these "ancestors". One way around this problem, is to use the construction of a so-called kernel set described in the Appendix. For a given derivation horizon and a given context, this construction inductively builds the set of minimal falsity implying subsets of statements in Γ. This kernel set can reasonably be said to contain all statements that are "to blame" for the inconsistency of the context (within the horizon), hence we can define safe contraction as the removal of all these statements and their dependents. Although this will yield a unique solution, it will usually not be minimal in terms of the number of statements that are removed.

8 Concluding remarks

Since its birth in 1903, type theory has proved to be a useful medium for the design and implementation of deductive systems, programming languages and theorem provers. This paper explored the use of type theory to provide a deductive approach to belief revision which can be easily implemented. The starting idea is that type theory enables explicit representations of justifications in belief revision. With the representation of beliefs as type theoretical statements and belief states as type theoretical contexts, we showed that the presence of justifications makes it easy to identify the beliefs that cause inconsistency of the belief state (section 4.2). Their presence also greatly simplifies the handling of dependencies between beliefs (section 5.1). With respect to literature, our initial assumptions put us in the area of foundations theory for belief bases. However, our account does not suffer from the drawbacks usually associated with foundations theory such as problems with disbelief propagation, circular justifications, and multiple justifications for the same belief (section 6.2). The operation of belief revision that naturally arises from our approach is one of non-prioritized revision: new information is not automatically completely trusted (section 7.3).

The fact that our approach is deductive, and that we do not require that our theory of belief revision itself selects which beliefs have to be removed, makes its applicable to agents with limited computational resources (see appendix). This holds independently of the strength of the logic in which the belief change operations are cast: the mechanisms that were used to represent justifications and dependency relations between beliefs are at the heart of type theory, making our approach applicable to a large family of type systems. Given the well established connections between type theory and logic, this means it is applicable in a wide range of logics. For instance, it can be applied in each of the Pure Type Systems from the well-known Logic Cube [4], which corresponds to logics ranging from minimal propositional logic to higher order predicate logic. Our immediate goal is to use the extensive research in implementations of logics based on type theory in order to provide a working automated system of belief revision based on the approach of this paper.

References

[1] Ahn, R., Borghuis, T., Communication Modelling and Context-Dependent Interpretation: an Integrated Approach. In: *Types for Proofs and Programs: International Workshop, TYPES'98*. Selected Papers, Altenkirch, T., Naraschewski, W., Reus, B. (eds.), LNCS 1657, Springer Verlag (1999), pp. 19 – 32.

[2] Alchourrón, C.E. Gärdenfors, P., and Makinson, D., On the logic of theory change: partial meet contraction and revision functions, *Journal of Symbolic Logic* 50 (1985), pp. 510-530.

[3] Alchourrón, and Makinson, D., On the logic of theory change: safe contraction, *Studia Logica* XLIV (1985), pp. 405-422.

[4] Barendregt, H., Lambda calculi with types. In *Handbook of Logic in Computer Science*, Abramsky, Gabbay and Maibaum (eds.), Oxford University Press, Oxford (1992), pp. 117 – 309.

[5] Barras, B., et al., The Coq Proof Assistant Reference Manual – Version V6.1, Technical report 0203, INRIA, 1997.

[6] J. van Benthem, Reflections on epistemic logic, *Logique & Analyse* 133-134 (1991), pp. 5-14.

[7] Berardi, S., Towards a mathematical analysis of the Coquand-Huet calculus of constructions and the other systems in Barendregt's cube. Technical report, Dept. of Computer Science, Carnegie-Mellon University and Dipartimento Matematica, Universita di Torino, 1988.

[8] Bruijn, N.G. de, AUTOMATH, a language for mathematics, Technical report 68-WSK-05, Technische Universiteit Eindhoven, 1968.

[9] Bunt, H., Ahn, R., Beun, R-J., Borghuis, T., and Van Overveld, K., Multimodal Cooperation with the DenK System. In: *Multimodal Human-Computer Interaction*, Bunt, H., Beun, R-J., Borghuis, T. (eds.), Lecture Notes in Artificial Intelligence 1374, Springer Verlag (1998), pp. 39 – 67.

[10] Chopra, S., Parikh, R. and Wasserman, R., Approximate belief revision, *Logic Journal of the IGPL*, Vol. 9 No. 6 (2001), pp. 755-768.

[11] Church, A., A formulation of the simple theory of types. *The Journal of Symbolic Logic*, 5:56–68, 1940.

[12] Coquand, T., and Huet, G., The calculus of constructions. *Information and Computation* 76, 95-120, 1988.

[13] Curry, H.B. and Feys, R., *Combinatory Logic I*, Studies in Logic and the Foundations of Mathematics, North-Holland, Amsterdam, 1958.

[14] Freund, M., Full meet revision on stratified bases, *Theoria* 67 (2001) nr. 3, pp. 189-213.

[15] Gabbay, D., *Labelled Deductive Systems.* Oxford University Press.

[16] Gabbay, D., Hunter, A., Making inconsistency respectable 1: a logical framework for inconsistency in reasoning. In: *Foundations of AI Research*, Ph. Jarrand and J. Keteman (Eds.) LNCS 535, pp. 19-32 (1991).

[17] Gärdenfors, P., The dynamics of belief systems: Foundations versus coherence theories, *Revue Internationale de Philosophie*, 44 (1990), pp. 24 – 46.

[18] Gärdenfors, P., Roth, H., Belief Revision. In: *Handbook of Logic in Artificial Intelligence and Logic Programming*, Volume 4: Epistemic and Temporal Reasoning, Gabbay, D., Hogger, C., Robinson, J. (eds.), Clarendon Press (1995), pp. 36 – 119.

[19] Hansson, S., In defense of base contraction, *Synthese* 91 (1992), pp. 239 – 245.

[20] Hansson, S., Taking belief bases seriously. In: *Logic and the Philosophy of Science in Uppsala*, Prawitz, D. and Westerståhl, D. (eds.), Kluwer Academic Publishers (1994), pp. 13 – 28.

[21] Hansson, S., Semi-revision, *Journal of Applied Non-Classical Logics*, Volume 7, nr. 1-2 (1997), pp. 151 – 175.

[22] Hansson, S.O. (Ed.), *Special issue on non-prioritized belief revision*, *Theoria* 53 (1997) Part 1-2, pp. 1-134.

[23] Harper, R. and Honsell, F. and Plotkin, G., A framework for defining logics, IEEE Proceedings Second Symposium on Logic in Computer Science, pages 194–204, 1987.

[24] Heyting, A., *Mathematische Grundlagenforschung. Intuitionismus. Beweistheorie.* Springer Verlag, Berlin, 1934.

[25] Howard, W.A., The formulas-as-types notion of construction, In To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism, editors Seldin, J.P. and Hindley, J.R., Pages 479–490, 1980. Academic Press, New York.

[26] Kamareddine, F., Bloo, R., and Nederpelt, R.P., On π-conversion in the λ-cube and the combination with abbreviations. *Annals of Pure and Applied Logics*, 97:27–45, 1999.

[27] F. Kamareddine and R.P. Nederpelt. Canonical typing and Π-conversion in the Barendregt Cube. *Journal of Functional Programming*, 6(2):245–267, 1996.

[28] Kolmogorov, A.N., Zur Deutung der Intuitionistischen Logik, *Mathematisches Zeitschrift*, Volme 35, Pages 58–65, 1932.

[29] Laan, T., *The Evolution of Type Theory in Logic and Mathematics*, PhD thesis, Technische Universiteit Eindhoven, 1997.

[30] Milner, M., Tofte, M., and Harper, R., *The Definition of Standard ML*. MIT Press, 1991, Cambridge, MA.

[31] Nederpelt, R.P., Geuvers, J.H., De Vrijer, R.C. (Eds.), *Selected Papers on Automath*, Studies in Logic and the Foundations of Mathematics 133, North-Holland, Amsterdam (1994).

[32] Tait, W.W., Infinitely long terms of transfinite type, In *Formal Systems and Recursive Functions*, Crossley, J.N. and Dummett, M.A.E., (editors), North-Holland, Amsterdam, 1965.

[33] J. Terlouw. Een nadere bewijstheoretische analyse van GSTT's. Technical report, Department of Computer Science, University of Nijmegen, 1989.

A Kernel consolidation

Our revision procedure is particularly close to what Hansson calls *kernel consolidation*. This form of consolidation is based on the idea that a subset of sentences in the knowledge base K implies falsity if and only if this subset contains some minimal falsity-implying subset of K. Hence the consistency of K can be restored by removing at least one element of each minimal falsity-implying subset of K. Minimal falsity-implying subsets are called *kernels*, they are defined as follows.

Definition A.1
A subset X of sentences from a belief base K is a *kernel* if:
1. $X \subseteq K$
2. $\bot \in Cn(X)$, and
3. If $Y \subset X$, then $\bot \notin Cn(Y)$

The set of all kernels of K is called the *kernel set*, denoted by $K \coprod \bot$.

The sentences of K that have to be discarded to restore consistency, are selected by an *incision function*:

Definition A.2
An *incision function* σ for K is a function such that:
1. $\sigma(K \coprod \bot) \subseteq \cup(K \coprod \bot)$
2. If $X \in (K \coprod \bot)$, then $X \cap \sigma(K \coprod \bot) \neq \emptyset$

Definition A.3
Let σ be an incision function for K. The *kernel consolidation* $\approx_\sigma$ for K is defined as follows:

$$K \approx_\sigma \bot = K \backslash \sigma(K \coprod \bot)$$

In the typetheoretical approach, falsity-implying subsets of the context Γ are sets of statements of which the subjects occur free in a proof object inhabiting $\bot$, i.e. $\{\mathtt{stat}_\Gamma(y) | y \in FV(M)\}$, where M is a term such that $\Gamma \vdash M : \bot$. If we call this set of statements for a given proof object M 'S^M' ('suspects' in M), we can see that this set fulfils the first two criteria for kernels given in Definition A.1:
1. $S^M \subseteq \Gamma$
2. $\Gamma_{init}, S^M \vdash M : \bot$, that is: $\bot$ is a consequence of S^M (where Γ_{init} contains the well-typedness information needed for the derivation)

However, such a falsity-implying subset S^M is not necessarily minimal in the sense required for kernels (the third criterion): there may exist another proof object N such that $\Gamma \vdash N : \bot$ and $S^N \subset S^M$. This is due to the fact that proof objects code an entire proof for the proposition represented by their type, including proofs that contain 'detours', sequences of steps that could have been omitted in the proof. Such detours can invoke premises that are not really needed to prove the proposition, resulting in non-minimal subsets. A very simple example of this is the following: take $\Gamma \equiv \Gamma_{init}, x : A, z : A \to A, y : A \to \bot$, then there are at least two proof objects inhabiting falsity, $\Gamma \vdash y(zx) : \bot$ and $\Gamma \vdash yx : \bot$. Clearly, the falsity-implying subset for the first proof object is not minimal, the second proof object is constructed without using $z : A \to A$. Although in typed λ-calculus some detours can be eliminated by performing reductions on proof objects[5], we cannot in general prevent a person from having a belief state on which non-minimal proofs of falsity can be derived.

Moreover, in discussing the minimality of falsity-implying subsets, the limited deductive powers have to be taken into account. Since the person can only construct proofs of $\leq h$ steps, where h is the horizon distance, we can at best talk about falsity-implying subsets which are minimal with respect to these proofs. Given a subset S^M for some inhabitant M of falsity, there may exist a set S^N such that $S^N \subset S^M$ where the proof object N for falsity cannot be constructed within the horizon h. Hence, this smaller set S^N should not be considered by the selection procedure.

The assumption of horizon enables an inductive procedure for the constructing the kernel set $\Gamma \coprod^h \bot$, the set of all minimal falsity-implying subsets within the horizon. For a given context Γ, one systematically generates all derivations of length zero,then all derivations of length 1, then all derivations of length 2, ..., up to all derivations of length h. Among each layer of derivations, one picks out all derivations of an inhabitant of falsity. By comparing the sets of free variables of these inhabitants, the minimal falsity-implying subsets for that layer can be found, i.e. for the i-th layer ($1 \leq i \leq h$) all $FV(M)$ such that $\Gamma \vdash^i M : \bot$, and there is no N such that $\Gamma \vdash^i N : \bot$ and $FV(N) \subset FV(M)$. The sets S^M that are minimal for a layer are then added to the kernel set $\Gamma \coprod^i \bot$ if there is no S^N already in $\Gamma \coprod^i \bot$ such that $S^N \subset S^M$. In other words, before adding the sets that are minimal in a layer it is a checked whether they are also minimal with respect to sets from previous layers.

Given the inductively constructed kernel set $\Gamma \coprod^h \bot$, the type theoretical analogons of *incision function* and *kernel consolidation* can be defined exctly as given in Definitions A.2 and A.3, but for the replacement of $K \coprod \bot$ by $\Gamma \coprod^h \bot$. Note that in the newly attained definition the slash in $\Gamma \backslash \sigma(\Gamma \coprod^h \bot)$ stands for the type theoretical removal operation defined in section 5.1, rather than the standard set theoretical operation in definition A.2, i.e. not only the statements selected by the incision function ($\sigma(\Gamma \coprod^h \bot)$) are removed from Γ but also all statements depending on them ($\mathtt{dep}_\Gamma(\sigma(\Gamma \coprod^h \bot))$). Since dependencies are not considered in the setting of Hansson, we need to be able to distinguish between those two kinds of statements. The notion of 'independence' can easily be defined as follows:

Definition A.4
A statement $x := M : A$ is an *independent member* of the set of statements Δ iff there is no statement $z := N : B \in \Delta$ such that $x \in \mathtt{dep}_\Delta(z)$.

In [21], kernel consolation is characterised by a theorem linking its construction to a number of postulates. We restate this theorem for type theoretical knowledge states:

Theorem A.5
An operation $>$ defined on type-theoretical knowledge states is an operation of kernel consolation iff for all contexts Γ:

1. $(\Gamma >)$ is consistent *(consistency)*
2. $(\Gamma >) \subseteq \Gamma$ *(inclusion)*
3. If $x := M : A$ is an independent member of $\Gamma - (\Gamma >)$, then there is Γ' such that $\Gamma' \subseteq \Gamma$, Γ' is consistent and $\Gamma', x := M : A$ is inconsistent *(core-retainment)*.

Proof. As x is independent, the proof is analogous to that of Hansson. There are two cases in the proof where the independence is needed to ensure that a statement is an element of $\sigma(\Gamma \coprod^h \bot)$ rather than merely an element of $\mathtt{dep}_\Gamma(\sigma(\Gamma \coprod^h \bot))$: in proving core-retainment in the direction

[5]Sometimes a term representing a non-minimal proof can be β-reduced to a minimal one, since β- reduction corresponds to cut elimination: take $\Gamma \equiv \Gamma_{init}, x : A, y : B, z : A \to \bot$, and $M \equiv (((\lambda u : A.(\lambda v : B.u))x)y)z : \bot$, then the $\bot$-implying subset S^M is $\{x : A, y : B, z : A \to \bot\}$. After performing β-reduction twice, we find the normal form of M which is xz. Now $\{x : A, z : A \to \bot\}$ is a minimal $\bot$-implying subset.

from construction to postulates, and in proving that σ is an incision function in the direction *from postulates to construction.* ∎

Received 10 February 2002