ISTITUTO PER LA RICERCA SCIENTIFICA E TECNOLOGICA
I 38100 TRENTO – LOC. PANTÉ DI POVO – TEL. 0461–314444
TELEX 400874 ITCRST – TELEFAX 0461–302040

THE INEVITABILITY OF INCONSISTENT ABSTRACT SPACES

F. Giunchiglia, T. Walsh

June 1990
Technical Report # 9006-16

Published in *Journal of Automated Reasoning*, Vol. 11, pp. 23–41, 1993.

ISTITUTO TRENTINO DI CULTURA

The Inevitability of Inconsistent Abstract Spaces *

Fausto Giunchiglia
Mechanized Reasoning Group
IRST, Povo, 38100 Trento, Italy
DIST, University of Genoa, Genoa, Italy
fausto@irst.it

Toby Walsh
Department of Artificial Intelligence
University of Edinburgh
80 South Bridge, EH1 1HN Edinburgh, Scotland
T.Walsh@uk.ac.edinburgh

September 8, 1992

Abstract

Abstraction has been widely used in automated deduction; a major problem with its use is that the abstract space can be inconsistent even though the ground space is consistent. We show that, under certain very weak conditions true of practically all the abstractions used in the past (but true also of a much wider class of abstractions), this problem cannot be avoided.

*This work was begun when the first author was at the AI Department of Edinburgh University. In Edinburgh, financial support to the first author was by SERC grant GR/E/4459.8. Financial support to the second author is provided by a SERC PostDoctoral Fellowship. The research described in this paper owes a lot to the openess and sharing of ideas which exists in the Mathematical Reasoning group in Edinburgh and Mechanized Reasoning group in Trento. In particular the authors thank Alan Bundy, Alessandro Cimatti, Craig Knoblock, Luciano Serafini, Alex Simpson, for their comments and for their careful reading of many earlier versions of this paper.

1 Introduction

Abstraction has been frequently used in Artificial Intelligence and Automated Deduction (*eg.* problem solving [NSS63], planning [Sac74], theorem proving and logic programming [Plu87, Pla81], cognitive modelling/ learning [UR89]), within many different formal systems (*eg.* production systems, planning systems, natural deduction systems, refutation systems). In [GW89a, GW92] we provide a formal framework where all these different applications are interpreted uniformly.

The central idea underlying the use of abstraction is that of mapping a representation of a problem, the "*ground*" representation, onto a new representation, the "*abstract*" representation; this mapping helps solve the problem in the original search space by preserving certain desirable properties and by being simpler to handle. In most cases the goal is to prove the abstracted theorem and then to use the structure of this proof to help construct a proof of the original theorem. This relies upon the assumption that the structure of the two proofs is "similar". The class of abstractions we concentrate on in this paper corresponds exactly to those used in this way. In [GW92] this and other uses of abstraction are studied in detail.

Various researchers have noticed that the abstraction of a consistent set of ground axioms can be inconsistent [Nil80, Pla80, Ten87]. We call this the *"problem of inconsistent abstract spaces"*. The aim of this paper is:

- to understand what causes the generation of inconsistent abstract spaces. We show that inconsistent abstract spaces are generated because the set of theorems of the abstract space is a strict superset of the abstraction of the set of theorems of the ground space.
- to study to what extent the problem can be avoided. We show that it is impossible to guarantee in advance the consistency of the abstract space. There will always exist a consistent set of axioms whose abstraction is inconsistent. As a consequence the problem of inconsistent abstract spaces must be dealt with at runtime.

Section 2 informally describes the problem. Section 3 introduces the theory, whilst section 4 gives the main results. We end in section 5 with some concluding remarks.

2 The Problem

Intuitively, the problem of generating an inconsistent abstract space can be described as follows. Let us suppose we have a set of axioms which constitute the ground theory. For instance, let us suppose that we have a very simple theory which has

only one axiom, namely the formula $P(a) \wedge \neg P(b)$. This theory is consistent; that is, it has a model. Now let us suppose that our abstraction deletes all the arguments from the predicate symbols transforming them into propositional constants. In this example the result is the formula $P \wedge \neg P$ which is clearly inconsistent (provided that negation and conjunction have their "usual meaning"). The informal explanation of why this happens is as follows. In order to build the abstract space "simpler" than the ground space, we needed to "forget" some "irrelevant" details, keeping around those details that were judged important. The problem is that the irrelevant looking details may be exactly those that are preserving the theory from being inconsistent, making them rather relevant.

This problem occurs with the abstraction used in Abstrips which deletes preconditions to Strips operators [Sac74, GW92]. Nilsson, for example, in figure 8.13 on page 353 of [Nil80], gives an example where the contradictory facts $ON(A, C)$ and $ON(C, A)$ both hold in the abstract space. The problem of inconsistent abstract spaces was also noticed by Plaisted [Pla80, Pla81] and Tenenberg [Ten87, Ten88]. In one example, Tenenberg shows how collapsing two predicate names together can give an inconsistent abstract space. For example, the axiom $P(a) \wedge \neg Q(a)$ maps onto $R(a) \wedge \neg R(a)$ if the predicate names, P and Q are both abstracted onto R. Both Tenenberg and Plaisted worked in a refutation system; thus the problem is not an inconsistent abstract space (which a refutation system seeks when trying to prove an abstracted goal) but rather that the abstraction of the axioms without the negated goal is inconsistent. In such a situation, any abstracted goal can be proven. Worse still, the structure of the abstract proof will often have nothing informative to say.

But is this really so bad? The answer is not immediately obvious. For instance Nilsson, in page 352 of [Nil80], argues that "*... A contradictory state description may result, but this causes no problems. ...*". Tenenberg in page 39 of [Ten88] argues the contrary, "*... once such a situation is reached, there are no constraints on the future choice of actions. ...*".

There is a point behind Nilsson's statement as it is not always worthwhile worrying about the consistency of the abstract space. In general, testing consistency is neither decidable nor tractable. Even if the abstract space is inconsistent, the structure of the abstract proof could still be used to guide the search for a proof in the ground space. After all, we are always working with finite resources and many of the branches of the proof tree do not use the fact that the theory is inconsistent (*eg.* they do not use the fact that $\perp$ is a theorem and that anything can be derived from $\perp$). If the procedure for mapping abstract proofs back onto ground proofs does not use much information from the abstract space the chances of using inconsistent information are slight. This is what happens in Nilsson's example: no further abstract spaces are generated from the inconsistent abstract space and extensive reasoning is not performed in the abstract space.

Besides the theoretical issue, which we think is of substantial importance, we would argue that this problem is very important for any implementation that uses abstraction. The inconsistency of the abstract space is always a bad thing as it means that something has gone wrong. The decision to worry about this problem, or to ignore it, should be made at runtime, on a case by case basis. It should not be an assumption made in designing the system. Of course, there is in general no guarantee that abstract proofs will map back onto ground proofs. However, abstract proofs which exploit the inconsistency of the abstract space will never map back. If theorem proving in the abstract space is not suitably restrained, inefficiency will be introduced by such false attempts (what Plaisted called "false proofs" [Pla81]); this could make reasoning with abstraction less efficient than reasoning without abstraction. If arbitrary theorem proving is allowed in the inconsistent abstract space, any abstract formula is a theorem; none of the formulae which are not theorems in the ground space are therefore filtered out. For this reason, theorem proving in an inconsistent abstract space in general provides too little information.

3 A Theory of Abstraction

The aim of this section is to provide a formal description of abstraction. The definitions introduced here characterise all previous abstractions of which we are aware, and give general properties which should be satisfied by any "useful" abstraction. In [GW89a], we show how Abstrips [Sac74], Plaisted's ordinary and weak abstractions [Pla80], Hobb's granularity [Hob85] and many other types of abstraction fit into this framework. Many more examples are described in [GW92].

3.1 The basic concepts

An abstraction is defined as a mapping between the representations of two problems. We formalise the notion of "representation of a problem" as an axiomatic formal system. In other words, we take it to be a set of known facts, Ω (the axioms) written in a certain language, Λ (which contains all the well formed formulas) plus a set of inference rules, Δ (the deductive machinery) which allows us to derive new facts from old.

Definition 1 (Formal system) *: A formal system Σ is a triple $\langle \Lambda, \Delta, \Omega \rangle$, where Λ is the language, Ω is the set of axioms and Δ is the set of inference rules, called the deductive machinery, of Σ.*

For the sake of simplicity we restrict ourselves to logical first order systems, where the logical axioms and logical inference rules are those of any formal system complete for classical first order logic, *eg.* natural deduction, resolution. Even if the results

presented in the following can be generalised to some significant subsets of first order logic, *eg.* propositional calculus, intuitionistic logic, we will not deal with this issue in this paper. As a further simplification we consider those formal systems with no logical axioms and no theoretic inference rules; the only possible axioms are the theoretic ones while the only possible inference rules are the logical ones. This means, for instance, that we do not consider Hilbert systems (like those described in [Men64]) nor systems where the theory is given in terms of inference rules (*eg.* Peano arithmetics, as given in [Pra71]). These last restrictions are not constraining as we still capture all the work in abstraction. In addition, they could be lifted by making proofs technically more complicated.

We can now define an abstraction as a mapping between formal systems.

Definition 2 (Abstraction) *: If* $\Sigma_1 = \langle \Lambda_1, \Omega_1, \Delta_1 \rangle$ *and* $\Sigma_2 = \langle \Lambda_2, \Omega_2, \Delta_2 \rangle$ *are two formal systems, an abstraction* f *is a pair* $\langle \Sigma_1, \Sigma_2 \rangle$ *and a triple of effective, total and surjective functions* $\langle f_\Lambda, f_\Omega, f_\Delta \rangle$ *such that:*

$$f_\Lambda : \Lambda_1 \to \Lambda_2$$
$$f_\Omega : \Omega_1 \to \Omega_2$$
$$f_\Delta : \Delta_1 \to \Delta_2$$

Following historical convention [Sac74], we call Σ_1 the ground space and Σ_2 the abstract space. The idea behind the definition of abstraction is that, in abstracting a representation of a problem, we chose how to abstract the language, the axioms and the inference rules. Effectiveness of the mapping functions is required to guarantee that the abstract space is generated in a finite amount of time. Totality allows us to translate any well formed formula (wff, from now on), axiom or inference rule of the ground space into the abstract space. Surjectivity guarantees that the abstract space is completely defined by the ground space and the abstraction mapping. This hypothesis is always satisfied by the abstractions described in the introduction; moreover it is necessary to make proofs go through. For the sake of brevity, and wherever there is no ambiguity, we will write $f(\alpha)$ instead of $f_\Lambda(\alpha)$.

The use of abstraction considered here is to build an abstract proof and then to map this proof back onto a proof in the ground space. Such abstractions should preserve completeness: if a wff is a theorem of the ground space, then its abstraction is a theorem of the abstract space. We write $\mathrm{TH}(\Sigma)$ to mean the set of theorems of Σ, *ie.* the minimal set of wffs containing the axioms and closed under the inference rules.

Definition 3 (TI abstractions) *: An Abstraction* $f : \Sigma_1 \Rightarrow \Sigma_2$ *is said to be a TI Abstraction iff, for any wff* α*, if* $\alpha \in TH(\Sigma_1)$ *then* $f_\Lambda(\alpha) \in TH(\Sigma_2)$*. TI Abstractions are also called truthful abstractions.*

"TI" means "Theorem Increasing". Provability is a minimal property which an abstraction should preserve; there is no reason to make your abstract theorem prover incomplete and certainly not in an uncontrolled fashion. Note that $TH(\Sigma_2)$ contains at least the abstraction of all the theorems of Σ_1. When $TH(\Sigma_2)$ contains just the abstraction of the theorems of Σ_1 we say that the abstraction is "TC" (meaning "Theorem Constant"). Nearly all the abstractions defined in the past are TI and not TC.

Figure 1 describes graphically the behaviour of TI abstractions. In this and the following figure the two boxes represent the sets of wffs belonging to the two languages. The dashed lines show the behaviour of the abstraction mapping. If no dashed lines are shown, then there is no restriction on how a subset of the ground language is mapped into the respective subset of the abstract language. The dimensions of boxes and circles are irrelevant.

3.2 An Example

As an example, we give below a rational reconstruction of the Abstrips abstraction, originally proposed in [Sac74] and proved to be TI in [GW92]. This abstraction has been chosen for many reasons. First, it is of historical interest, being one of the first abstractions ever proposed. Second, it is one the most widely used in problem solving. Finally, if applied in refutation systems (see next section), it is one of the few interesting abstractions which does not fall into any of the classes of abstractions proposed by Plaisted [Pla80, Pla86, Pla81]. The description given here follows closely that originally presented in [GW89a] and then extended in [GW92].

Abstrips built plans in which operators applied to states of the world and generated new states. Operators had preconditions, consisting of atomic formulae which had to be satisfied for them to be applied. Preconditions were abstracted according to a natural number associated to them, called *criticality*. The idea was to build a hierarchy of abstract spaces, each level in the hierarchy containing all and only the preconditions above a given criticality. We write that $i \in crit(\kappa)$ to mean that the criticality of the precondition p_i is greater than κ.

To put Abstrips into a theorem proving context we use the situation calculus notation [MH69] and follow Green's "logical reconstruction" of Strips planning [Gre69]. The ground space is therefore a situation calculus with a first order language, frame, operator and theoretic axioms. Operators are wffs of the form " $\forall s.\ (\bigwedge_{1 \leq i \leq n} p_i(s) \rightarrow\ q(f(s)))$ " where p_i is a precondition, s is a situation, f is some action, and q describes the new situation. Goals are wffs of the form "$\exists s\ r(s)$".

We can now formalize Abstrips as an abstraction, $f_{AB} : \Sigma_1 \Rightarrow \Sigma_2$ where both Σ_1 and Σ_2 are first order calculi using some set of inference rules complete for first order clasical logic, *eg.* natural deduction. The mapping on the inference rules, f_Δ

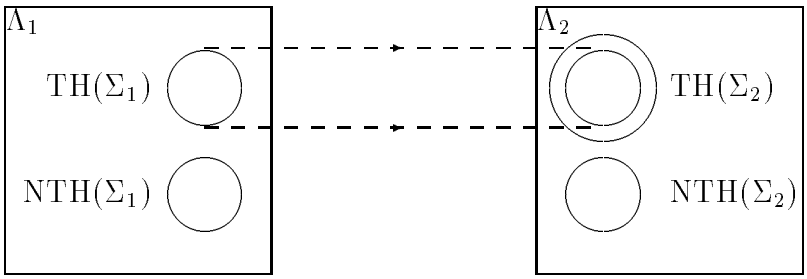

Figure 1: TI abstractions (Truthful abstractions)

becomes the identity mapping. f_Λ is defined as follows:

$f_{AB}(\alpha) = \alpha$, if α is atomic;

$f_{AB}(\neg\alpha) = \neg f_{AB}(\alpha)$;

$f_{AB}(\alpha \circ \beta) = f_{AB}(\alpha) \circ f_{AB}(\beta)$, where "$\circ$" is "$\wedge$" or "$\vee$";

$f_{AB}(\sharp x.\alpha) = \sharp x.f_{AB}(\alpha)$, where "$\sharp$" is "$\exists$" or "$\forall$";

$f_{AB}(\alpha \rightarrow \beta) = f_{AB}(\alpha) \rightarrow f_{AB}(\beta)$, if "$\alpha \rightarrow \beta$" is not an operator;

$f_{AB}(\bigwedge_{1 \leq i \leq n} p_i(s) \ \rightarrow \ r) = \bigwedge_{i \in crit(\kappa)} p_i(s) \ \rightarrow \ f_{AB}(r)$, for any operator.

f_Ω is defined to work the same as f_Λ on the axioms.

For example, the operator for going through a door

$$at(z, x, s) \wedge open(door, z, s) \rightarrow at(z, x, gothrough(door, z, s)) \tag{1}$$

might abstract to one in which we do not bother to check that the door is open, that is

$$at(z, x, s) \rightarrow at(z, x, gothrough(door, z, s)) \tag{2}$$

3.3 Preserving Inconsistency

If we are also to capture the use of abstraction in refutation systems, we need to introduce a new and apparently different class of abstractions; this class of abstractions maps inconsistent systems onto inconsistent systems preserving completeness. An important distinction is between absolute inconsistency (a theory is absolutely inconsistent iff any wff is provable) and inconsistency (a theory is inconsistent iff there exist a wff α such that both α and $\neg\alpha$ are provable). The problem of inconsistent abstract spaces therefore arises when the abstract space is absolutely inconsistent. In classical logic (and in all the cases considered here, see lemma 1 below) the two concepts coincide.

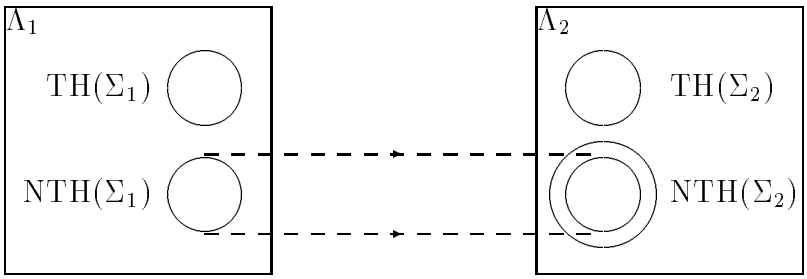

Figure 2: NTI abstractions (Falseful abstractions)

We write NTH(Σ) to mean the minimal set of all the wffs, such that, if added as an assumption, make Σ inconsistent. The elements of $NTH(\Sigma)$ are called nontheorems. (Notice that assumptions are not axioms, the difference affecting the proof theory only in the case of open formulas. For instance, assumptions, differently from axioms, in natural deduction may prevent the application of forall introduction and exists elimination [Pra65]. In Hilbert calculi they may cause the non applicability of the deduction theorem [Men64]. In resolution the distinction between axioms and assumptions is irrelevant. As the mapping of the set $NTH(\Sigma)$ is important only in the case of refutation systems, we will use the word "axiom" and the word "assumption" synonymously.) $TH(\Sigma)$ and $NTH(\Sigma)$ are obviously related. For instance in classical first order logics, $\alpha \in NTH(\Sigma)$ iff $\neg\alpha \in TH(\Sigma)$; Σ is inconsistent iff $TH(\Sigma) = NTH(\Sigma) = \Lambda_\Sigma$. Entirely dual to the class of TI abstractions is thus the class of NTI abstractions.

Definition 4 (NTI abstractions) *: An Abstraction $f : \Sigma_1 \Rightarrow \Sigma_2$ is said to be an NTI Abstraction iff, for any wff α, if $\alpha \in NTH(\Sigma_1)$ then $f_\Lambda(\alpha) \in NTH(\Sigma_2)$. NTI abstractions are also called falseful abstractions.*

"NTI" means "NonTheorem Increasing". Figure 2 describes graphically the behaviour of NTI abstractions. We use the notion of NTC abstraction analogously to that of TC abstractions.

TI and NTI abstractions (from now on we write TI* abstractions to mean either class) coincide under some very weak assumptions captured by the following two definitions.

Definition 5 (System with negation) *: Σ is a formal system with negation iff its language contains negation (written "$\neg$") and is such that, for any expression α,*

1. *α is a wff iff $\neg\alpha$ is wff;*
2. *$\alpha \in TH(\Sigma)$ iff $\neg\alpha \in NTH(\Sigma)$;*
3. *$\neg\alpha \in TH(\Sigma)$ iff $\alpha \in NTH(\Sigma)$.*

All the most common first order systems (*eg.* Hilbert systems, all the ND calculi, resolution) are systems with negation. Note that for any "reasonable" system, conditions (1) and (2) imply that negation is classical and that, even for intuitionistic negation, condition (2) implies condition (3).

A formal system with negation gives negation its "usual" meaning. A key property is that if a system with negation is inconsistent then it is absolutely inconsistent.

Lemma 1 *: If a system with negation is inconsistent then it is absolutely inconsistent.*

Proof: If Σ is inconsistent then there exists a wff α such that $\alpha \in TH(\Sigma)$ ($\vdash_\Sigma \alpha$) and $\neg\alpha \in TH(\Sigma)$ ($\vdash_\Sigma \neg\alpha$). This means, by monotonicity, that for all wffs β, $\neg\beta \vdash_\Sigma \alpha$ and $\neg\beta \vdash_\Sigma \neg\alpha$. Then for all β, $\neg\beta \in NTH(\Sigma)$. Then, from condition 2 of definition 5 we have that for all wffs β, $\beta \in TH(\Sigma)$.

□

Definition 6 (Negation preserving abstractions) *: Let Σ_1 and Σ_2 be two systems such that α is a wff iff $\neg\alpha$ is. An abstraction $f : \Sigma_1 \Rightarrow \Sigma_2$ is negation preserving iff, for any α, $f(\neg\alpha) = \neg f(\alpha)$.*

The concept of a negation preserving abstractions between systems with negation allows us to bridge the gap between TI and NTI abstractions.

Theorem 1 *: If Σ_1, Σ_2 are two formal systems with negation, then any negation preserving abstraction $f : \Sigma_1 \Rightarrow \Sigma_2$ is a TI abstraction iff it is an NTI abstraction.*

Proof: We just consider the forward direction, the other direction is analogous. Since Σ_1 is a system with negation, if $\alpha \in NTH(\Sigma_1)$ then $\neg\alpha \in TH(\Sigma_1)$. But f is TI. Thus $f(\neg\alpha) \in TH(\Sigma_2)$. As f is negation preserving, $\neg f(\alpha) \in TH(\Sigma_2)$. From which it follows that $f(\alpha) \in NTH(\Sigma_2)$ and that f is NTI. □

The Abstrips abstraction given above is a negation preserving abstraction between systems with negation. (Negation preserving) TI* abstractions are a very wide class of abstractions. The only requirement they have is one of completeness: if a wff is provable (causes inconsistency) in the ground space then so must its abstraction; this is a very weak requirement. There are many other desirable requirements we may place on an abstraction. For example, there may be constraints on the mapping functions or on the structure of the proofs [GW92]. Earlier on we said that the inconsistent abstract spaces are generated because some relevant details are forgotten. Using this theory of abstraction, we will show that what matters is only the

impact an abstraction has on provability (or, for refutation systems, inconsistency). Under very weak assumptions, verified by all the abstractions we are aware of, the problem of inconsistent abstract spaces cannot be avoided as long as we work with TI* abstractions. First, however, we consider how to cope with the fact that the axioms are often not fixed in advance.

3.4 Independence of the Axioms

By defining an abstraction as a pair of formal systems with some mapping functions between them, we fix the ground and abstract spaces. However, in many applications the set of inference rules of the ground space are fixed but the axioms and language may vary, depending on the problem. In these cases it is possible to identify the following two steps:

- In the first (*off-line*) step, $\langle f_\Lambda, f_\Omega, f_\Delta \rangle$ are defined sufficiently general so that they will work for any expected choice of the language and axioms. This can be achieved by insisting that f_Λ and f_Ω are defined for the most general formal system, such that any formal systems we want can be obtained by suitably restricting the axioms and language.
- In the second (*runtime*) step, the user applies the abstraction to the particular problem to be solved by generating the abstract space. This can be achieved by building a new abstraction f' obtained from f by restricting f_Λ and f_Ω to the particular formal system at hand. f_Δ is usually left unmodified; however, some of the inference rules may no longer be applicable in the (restricted) ground space.

To capture the idea of an "abstraction defined for the most general formal system" we introduce the notion of "Λ-abstraction":

Definition 7 (Λ-abstraction) *: Any abstraction $f : \Sigma_1 \Rightarrow \Sigma_2$ where Σ_1 is of the form $\Sigma_1 = \langle \Lambda, \Lambda, \Delta \rangle$ is a Λ-abstraction.*

Note that the axioms of the ground space of a Λ-abstraction are the language itself and therefore that the ground space is absolutely inconsistent. To formalise the construction of f' out of f, we introduce the notion of Ω-restriction. (If $f : \Theta_1 \rightarrow \Theta_2$ is a function, then if $\Theta \subseteq \Theta_1$, by "$f \uparrow \Theta$" we mean f restricted to apply to Θ.)

Definition 8 (Ω-restriction) *Let $\Sigma_1 = \langle \Lambda_1, \Omega_1, \Delta_1 \rangle$, $\Sigma_2 = \langle \Lambda_2, \Omega_2, \Delta_2 \rangle$ be formal systems and $f : \Sigma_1 \Rightarrow \Sigma_2$ be the abstraction $\langle f_\Lambda, f_\Omega, f_\Delta \rangle$. Let $\Sigma_1' = \langle \Lambda_1, \Omega_1', \Delta_1 \rangle$, $\Sigma_2' = \langle \Lambda_2, \Omega_2', \Delta_2 \rangle$ where $\Omega_1' \subseteq \Omega_1$ and $\Omega_2' \subseteq \Omega_2$. Then $f' : \Sigma_1' \Rightarrow \Sigma_2'$, an Ω-restriction of f, is the abstraction $\langle f_\Lambda, f_\Omega', f_\Delta \rangle$, where $f_\Omega' = f_\Omega \uparrow \Omega_1'$ and Ω_2' is the codomain of f_Ω'.*

In other words, the Ω-restriction of an abstraction $f = \langle f_\Lambda, f_\Omega, f_\Delta \rangle$ has the same mappings on the language and deductive machinery. Its mapping function on axioms, f'_Ω, however, is built so that its domain and codomain are a subset of the domain and codomain of f_Ω and that f'_Ω and f_Ω agree on all the values where f'_Ω is defined. As a trivial example consider the case where Σ_1 and Σ_2 are both Peano arithmetics and f_Λ, f_Ω and f_Δ are the identity function. Then we can construct an Ω-restriction f' simply by taking Σ'_1 and Σ'_2 to be Peano arithmetics without the axiom of induction and by taking f'_Ω to be defined on the new set of axioms.

Notice that f' is itself an abstraction. Notice also that f is an Ω-restriction of itself. We write $f' \subseteq f$ to mean that f' is an Ω-restriction of f.

The concept of an Ω-restriction captures the way abstractions are commonly used: by limiting the set of ground axioms and by then building the abstract space from them. In Σ'_1 (Σ'_2), the language and inference rules could be tailored to fit the axioms; such an operation is unnecessary both theoretically and implementationally since the useless parts need never be used.

4 The Inevitability of Inconsistency

4.1 Some preliminaries

As the abstract space is constructed by applying an abstraction to the ground space, abstraction can be used in two different ways:

1. given a particular ground space (or set of spaces), we choose the most suitable abstraction;
2. given a particular abstraction (or set of abstractions), we apply it (them) to whatever ground spaces happen to occur.

In the first case, the application is fixed in advance. This occurs, for instance, in many areas of mathematical reasoning, where the formal system can be set theory (*eg.* group theory, geometry, number theory and so on), see for instance [Plu87, GW89b]. In such circumstances, a possible solution to the problem of inconsistent abstract spaces is, given a particular ground space (or class of ground spaces), to find abstractions which are proved in advance to construct consistent abstract spaces. This solution is only of theoretical interest as there are few theories whose intrinsic interest justifies such a time consuming operation. Moreover it is not obvious that such abstractions are really that "suitable"; the choice of the abstraction depends often on the goal to prove and cannot be made in advance.

The second case is far more interesting since in most applications the user is left free to choose his own set of axioms. The system is equipped with abstraction(s) which work on any possible set of axioms; at runtime, an abstraction is applied to the given axioms to generate the abstract space. In the rest of this section, we will concentrate on this second case and prove that with TI*-abstractions it is impossible to avoid inconsistent abstract spaces. In fact, no matter which abstraction you chose, there will always be a set of axioms (formally, an Ω-restriction) such that the abstract space is inconsistent.

Before proving the main result we need to prove a lemma which guarantees us that the class of abstractions we consider is not empty.

Lemma 2 *: Let $\Sigma_1 = \langle \Lambda_1, \Lambda_1, \Delta_1 \rangle$ be a formal system. Then there exists an abstraction $f : \Sigma_1 \Rightarrow \Sigma_2$ such that:*

1. *all its Ω-restrictions are TI;*
2. *(at least) one Ω-restriction is not TC.*

Proof: The proof is given by constructing an Abstrips abstraction $f : \Sigma_1 \Rightarrow \Sigma_2$ along the lines of that described in subsection 3.2. We take Λ_1 and Λ_2 to be the language needed to write formula (1) in subsection 3.2. Δ_1 and Δ_2 are taken to be natural deduction. f_Λ (and therefore f_Ω) is defined as in subsection 3.2. Moreover, f_Λ is such that the operator described by formula (1) gets translated into the formula (2) and this is the only case where preconditions are dropped. This completely defines f_Λ.

Part 1: It is sufficient to show that the fact that an Ω-restriction of f is TI does not depend on the choice of the axioms.

Consider the Ω-restriction f_i obtained by considering a given $\Omega_{1i} \subseteq \Omega_1$. We prove that f_i is TI by showing that, given a deduction tree Π_1 of α, we can build a deduction tree Π_2 of $f(\alpha)$ discharging the abstraction of the same assumptions. $f(\beta)$ is therefore provable in Σ_2 any time β is provable in Σ_1.

The proof proceeds by induction on the depth of the ground deduction tree. In the base case, f applied to the single wff in the tree generates a valid deduction in the abstract space. This single wff can be an assumption or an axiom belonging to Ω_{1i}.

Let us now consider the step case. We use $f(\Pi)$ to mean the tree in Σ_2 constructed from a tree, Π in Σ_1. Any rule application that is not modus ponens involving an operator translates unmodified. For an operator application, the following transformation is performed:

$$\dfrac{\dfrac{\Pi}{\bigwedge_{1\leq i\leq n} p_i} \qquad \bigwedge_{1\leq i\leq n} p_i \;\rightarrow\; q}{q} \quad \Longrightarrow \quad \dfrac{f\left(\dfrac{\Pi}{\bigwedge_{1\leq i\leq n} p_i}\right)}{\bigwedge_{i\in crit(\kappa)} p_i} \qquad \bigwedge_{i\in crit(\kappa)} p_i \;\rightarrow\; fq)}{f(q)}$$

By the induction hypothesis, and the fact that $\bigwedge_{i\in crit(\kappa)} p_i$ follows from $\bigwedge_{1\leq i\leq n} p_i$ by a possibly empty sequence of applications of and elimination, this is a valid abstract deduction tree which discharges the (abstraction of the) same assumptions as the deduction tree in Σ_1.

As it does not depend on the particular Ω_{1i} we have chosen, this proof can be repeated for all the Ω-restrictions of f. All the Ω-restrictions of f are therefore TI.

Part 2: Consider the Ω-restriction obtained by taking Ω_1 to consist only of formula (1) in subsection 3.2 plus the axiom $at(z, x, s)$. Then, formula (2) and $at(z, x, s)$ are axioms of Ω_2. Therefore, $at(z, x, gothrough(door, z, s))$ is a theorem of Σ_2 but not of Σ_1.

□

Lemma 2 demonstrates that the class of the abstractions considered in our main theorem (see next section) is not empty. What is actually important is that lemma 2 is satisfied by almost all abstractions and, in particular, by all the most important defined in the past. That this is the case can be easily seen by looking at the examples described in [GW92] (where proofs very similar to that of lemma 2 are carried through for almost all the examples).

The first condition of lemma 2 is very important as it allows us to build abstractions which can be applied to any set of axioms without losing completeness. This is what allows us to pre-compile sets of abstractions inside a system.

The second condition is also important as it guarantees that the abstraction makes problem solving easier. Increasing the number of theorems is intrinsic to the way abstractions throw away details. An Ω-restriction is TC in only a few isolated cases. For instance, when the abstraction performs an identity map (*eg.* in Abstrips abstractions, deleting non-existing preconditions) or when the ground space is inconsistent (*eg.* Λ-abstractions which are TI are also TC). For a TC abstraction, the theorems of the abstract space are exactly the abstractions of the theorems of the ground space. Any abstract proof will map back in a proof of the ground theorem. TC abstractions are thus in general too strong; they do not give "simpler" proofs except in very special and limited cases. For instance, if $f : \Sigma_1 \Rightarrow \Sigma_2$ is a TC abstraction and Σ_1 is undecidable then Σ_2 cannot be decidable. Of course this does not mean that TC abstractions are useless; they are often useful, for instance, in changing the representation of a problem.

4.2 The main result

We now give our main result. This shows that inconsistent abstract spaces are inevitable whenever we use TI*-abstractions.

Theorem 2 : *Let* $\Sigma_1 = \langle \Lambda_1, \Lambda_1, \Delta_1 \rangle$ *and* $\Sigma_2 = \langle \Lambda_2, \Omega_2, \Delta_2 \rangle$ *be two formal systems with negation. Let* $f : \Sigma_1 \Rightarrow \Sigma_2$ *be a negation preserving* Λ*-abstraction such that all its negation preserving* Ω*-restrictions between systems with negation are TI and one among them is not TC. Then there exists an* Ω*-restriction* $f' : \Sigma_1' \Rightarrow \Sigma_2'$ *of* f *such that* $\Sigma_1' = \langle \Lambda_1, \Omega_1', \Delta_1 \rangle$ *is consistent but* $\Sigma_2' = \langle \Lambda_2, \Omega_2', \Delta_2 \rangle$ *is absolutely inconsistent.*

Proof: The main steps of the proof are as follows:

1. We consider a (consistent) system (Σ_1'') such that an appropriate Ω-restriction $f'' : \Sigma_1'' \Rightarrow \Sigma_2''$ is TI but not TC and show that it is a system with negation
2. We prove that Σ_2'', is a system with negation;
3. We prove that f'' is negation preserving;
4. If Σ_2'' is absolutely inconsistent we are done, otherwise we build Σ_1' out of Σ_1'' by adding an axiom, $\neg\varphi$ such that $\varphi \notin TH(\Sigma_1'')$ but $f(\varphi) \in TH(\Sigma_2'')$;
5. We then construct $f' : \Sigma_1' \Rightarrow \Sigma_2'$, an Ω-restriction of f, and show that Σ_2' is absolutely inconsistent.

STEP 1: Let's consider a formal system $\Sigma_1'' = \langle \Lambda_1, \Omega_1'', \Delta_1 \rangle$, $\Sigma_1'' \subseteq \Sigma_1$ such that the appropriate Ω-restriction f'' is TI but not TC. f'' and Σ_1'' exist because of lemma 2. Σ_1'' is consistent. If Σ_1'' were not then f'' would be TC. Σ_1'' is also a system with negation. In fact, the following general result holds: if Σ is a system with negation then any $\Sigma^* = \langle \Lambda, \Omega^*, \Delta \rangle$, $\Sigma^* \subseteq \Sigma$ is also a system with negation. The proof goes as follows. The meaning of negation is given by the fact that you can always express both a formula and its negation and by the fact that you can use all the relevant inference rules. As we preserve both Λ and Δ, we are guaranteed to have a system with negation.

STEP 2: Let us now consider the Ω-restriction of f, $f'' : \Sigma_1'' \Rightarrow \Sigma_2''$, $f'' = \langle f_\Lambda, f_\Omega'', f_\Delta \rangle$, $f_\Omega'' = f_\Omega \uparrow \Omega_1''$. Remember that all the components of f'' are surjective, and Σ_2'' is thus completely defined by Σ_1'' and f''.

Σ_2'' is also a system with negation. In fact, the following general result holds: if $\Sigma_1 = \langle \Lambda_1, \Omega_1, \Delta_1 \rangle$ and $\Sigma_2 = \langle \Lambda_2, \Omega_2, \Delta_2 \rangle$ are any two systems with negation, $f : \Sigma_1 \Rightarrow \Sigma_2$ is an abstraction, $f^* : \Sigma_1^* \Rightarrow \Sigma_2^*$, and $f^* \subseteq f$, then Σ_2^* is a system with negation. The proof goes as follows. As in step 1, $\Sigma_1^* \subseteq \Sigma_1$ is a system with

negation. Since $f^* \subseteq f$, then for any component c of Σ_1 (a wff, an axiom or an inference rule) which is also in Σ_1^* we have that $f^*(c) = f(c)$. This means that everything which concerns negation gets translated correctly in Σ_2^*. Since f^* and f are surjective and total, Σ_2^* contains everything that gives meaning to negation in Σ_2 and nothing more. Anything which is in Σ_1^* and does not concern negation can't be translated incorrectly, namely in a way to make Σ_2^* not a system with negation (for instance by adding a wff to $TH(\Sigma_2^*)$ and not adding its negation to $NTH(\Sigma_2^*)$) since the components of Σ_2^* are taken to be exactly the codomain of the mapping functions; we are thus prevented for adding any extra effects. If this did not happen, since $f^* \subseteq f$, and Σ_2 is a system of negation, there should be components outside f^* which fix everything wrong concerning negation. But this would contradict the assumption that all the parts of f concerning negation are also in f^*.

STEP 3: f'' is negation preserving. In fact, if $f : \Sigma_1 \Rightarrow \Sigma_2$ is a negation preserving abstraction then any abstraction $f^* : \Sigma_1^* \Rightarrow \Sigma_2^*$, such that $f^* \subseteq f$, is a negation preserving abstraction. The proof goes as follows; for Σ_1^* and Σ_2^*, α is a wff iff its negation is; this is a corollary of the arguments given in steps 1 and 2. f^*'s mapping on the language is just f_Λ. As abstraction functions are total and the language of Σ_1^* is precisely Λ_1, we have $f_\Lambda(\neg\alpha) = \neg f_\Lambda(\alpha)$ for any $\alpha \in \Lambda_1^*$.

STEP 4: Σ_2'' may be either consistent or inconsistent. If Σ_2'' is inconsistent we just let $\Sigma_1' = \Sigma_1''$, $\Sigma_2' = \Sigma_2''$ and $f' = f''$ and we are finished. Σ_1'' and Σ_2'' are systems with negation. As Σ_2'' is inconsistent it is also absolutely inconsistent (lemma 1). Σ_1'' is consistent by construction. f'' is a negation preserving Ω-restriction of f by construction. f'' is TI and not TC by step 1.

Let's suppose that Σ_2'' is consistent.

f'' is TI and not TC. Therefore there exists a wff $\varphi \in \Lambda_1$ such that $f_\Lambda(\varphi) \in TH(\Sigma_2'')$ but $\varphi \notin TH(\Sigma_1'')$. Thus $\{\neg f_\Lambda(\varphi)\} \cup \Omega_2''$ is inconsistent but $\{\neg\varphi\} \cup \Omega_1''$ is consistent.

We now define $\Sigma_1' = \langle \Lambda_1, \Omega_1', \Delta_1 \rangle$ and $\Sigma_2' = \langle \Lambda_2, \Omega_2', \Delta_2 \rangle$, with $\Omega_1' = \{\neg\varphi\} \cup \Omega_1''$, and $\Omega_2' = \{f_\Omega(\neg\varphi)\} \cup \Omega_2''$.

STEP 5: We define $f' : \Sigma_1' \Rightarrow \Sigma_2'$, $f' = \langle f_\Lambda, f_\Omega', f_\Delta \rangle$ such that:

- for any $\omega \in \Omega_1''$, $f_\Omega'(\omega) = f_\Omega''(\omega) = f_\Omega(\omega)$,
- $f_\Omega'(\neg\varphi) = f_\Omega(\neg\varphi)$

f' is exactly what we need. In fact, Σ_1' and Σ_2' are systems with negation (all the arguments given for Σ_1'' and Σ_2'' apply here as well). Σ_1' is consistent by construction. From its definition, f' is a negation preserving Ω-restriction of f. Σ_2' is inconsistent by construction. In fact, since f' is an Ω-restriction of f and TI, from $\neg\varphi \in TH(\Sigma_1')$ it follows that $f_\Lambda(\neg\varphi) \in TH(\Sigma_2')$. Since f' is a negation preserving Ω-restriction of f we have that $\neg f_\Lambda(\varphi) \in TH(\Sigma_2')$. But $f_\Lambda(\varphi) \in TH(\Sigma_2')$, since Σ_2' is a monotonic extension of Σ_2''. Σ_2' is thus inconsistent (both a formula and its negation are

derivable) and absolutely inconsistent (being a system with negation, lemma 1). □

It is interesting to notice that the proof of theorem 2 is very similar to the way Ω-restrictions are used within a real system; that is, we build the abstract space by applying the abstraction function to the ground space.

4.3 Further Results

Theorem 2 demonstrates the inevitability of inconsistent abstract spaces for TI abstractions. We can give identical results for NTI abstractions since any negation preserving TI abstraction between systems with negation is also a NTI abstraction (theorem 1). In fact, theorem 2 can be generalised to NTI abstractions between formal systems which are not systems with negation since, in refutation systems, negation is dealt with inside the system by preserving inconsistency. The proof of the inevitability of inconsistent abstract spaces for NTI abstractions is therefore simpler than the proof for TI abstractions as we need not show that negation is preserved by the mapping.

Theorem 3 *: Let* $\Sigma_1 = \langle \Lambda_1, \Lambda_1, \Delta_1 \rangle$ *and* $\Sigma_2 = \langle \Lambda_2, \Omega_2, \Delta_2 \rangle$ *be two formal systems and* $f : \Sigma_1 \Rightarrow \Sigma_2$ *a* Λ*-abstraction such that all its* Ω*-restrictions are NTI and one among them is not NTC. Then there exists an* Ω*-restriction* $f' : \Sigma'_1 \Rightarrow \Sigma'_2$ *of* f *such that* $\Sigma'_1 = \langle \Lambda_1, \Omega'_1, \Delta_1 \rangle$ *is consistent but* $\Sigma'_2 = \langle \Lambda_2, \Omega'_2, \Delta_2 \rangle$ *is absolutely inconsistent.*

Proof[Outline]: We can repeat all the steps of the proof of theorem 2 but without worrying about preserving negation, *ie.* without checking that the systems involved are with negation and that the functions involved are negation preserving. □

Theorems 2, and 3 tell us that it is impossible to build TI*-abstractions which are guaranteed a priori to give consistent abstract spaces. In order to avoid inconsistent abstract spaces, we might decide to drop the restriction on using TI*-abstractions. Tenenberg advocates such a change in [Ten87]. He proposes a class of abstractions which are not TI* and are guaranteed to generate consistent abstract spaces. The problem with these abstractions (and all abstractions which are not TI) is that completeness is lost, *ie.* there are theorems of the ground space which are not theorems of the abstract space. When the abstract space is used to find a proof in the ground space, we consider completeness one property that you do not want to loose. Of course abstractions can be used in other ways; for instance they can be used to implement derived inference rules [GG88, GW92], but in these cases, to

retain completeness, the overall strategy of the theorem prover should be different and not inside the usual "abstraction" tradition.

Even accepting the restriction to TI*-abstractions, the request that all the Ω-restrictions which have a consistent ground space be TI* may seem to be too restrictive (even if lemma 2 and the following discussion should have convinced the reader that it is not). This restriction can actually be replaced by a weaker restriction. To this end, we will introduce a new notion which shows how the wffs and axioms of the ground and abstract spaces are abstracted.

Definition 9 (Λ/Ω-invariant abstraction) *: Let $\Sigma_1 = \langle \Lambda_1, \Omega_1, \Delta_1 \rangle$ be a formal system and $f : \langle f_\Lambda, f_\Omega, f_\Delta \rangle$, $f : \Sigma_1 \Rightarrow \Sigma_2$ be an abstraction, f is said to be Λ/Ω-invariant iff f_Ω is such that $f_\Omega(\omega_i) = f_\Lambda(\omega_i)$, for any $\omega_i \in \Omega_1$, Λ/Ω-variant otherwise.*

An abstraction is Λ/Ω-invariant iff the axioms are mapped in the same way as the wffs. All the abstractions we know of (except the one given in [Ten87]) are Λ/Ω-invariant. The Abstrips abstraction defined in subsection 3.2 is Λ/Ω-invariant. Λ/Ω-invariant abstractions are used whenever we do not distinguish between wffs and axioms. This is often the case when the abstraction is not tuned to a particular theory or when no special constraints are imposed on the abstract space. With Λ/Ω-invariant abstractions we can now drop the requirement of theorems 2, and 3 that all the Ω-restrictions are TI abstractions. Instead we require that at least one Ω-restriction is TI but not TC.

Corollary 1 *Let $\Sigma_1 = \langle \Lambda_1, \Lambda_1, \Delta_1 \rangle$ and $\Sigma_2 = \langle \Lambda_2, \Omega_2, \Delta_2 \rangle$ be two formal systems with negation. Let $f : \Sigma_1 \Rightarrow \Sigma_2$ be a negation preserving Λ/Ω-invariant abstraction such that there exists a negation preserving Ω-restriction between systems with negation which is TI but not TC. Then there exists a negation preserving Ω-restriction $f' : \Sigma'_1 \Rightarrow \Sigma'_2$ of f such that $\Sigma'_1 = \langle \Lambda_1, \Omega'_1, \Delta_1 \rangle$ is consistent but $\Sigma'_2 = \langle \Lambda_2, \Omega'_2, \Delta_2 \rangle$ is absolutely inconsistent.*

Corollary 2 *: Let $\Sigma_1 = \langle \Lambda_1, \Lambda_1, \Delta_1 \rangle$ and $\Sigma_2 = \langle \Lambda_2, \Omega_2, \Delta_2 \rangle$ be two formal systems and $f : \Sigma_1 \Rightarrow \Sigma_2$ be a Λ/Ω-invariant abstraction such that there exists an Ω-restriction which is NTI but not NTC. Then there exists an Ω-restriction $f' : \Sigma'_1 \Rightarrow \Sigma'_2$ of f such that $\Sigma'_1 = \langle \Lambda_1, \Omega'_1, \Delta_1 \rangle$ is consistent but $\Sigma'_2 = \langle \Lambda_2, \Omega'_2, \Delta_2 \rangle$ is absolutely inconsistent.*

Proof[corollary 1. Outline]: The proof is almost identical to that of theorem 2. Notice that all the Ω-restrictions of a Λ/Ω-invariant abstraction are also Λ/Ω-invariant. The main difference is that, once we have found the φ that is consistent with the axioms of Σ_1 but whose abstraction is inconsistent with the axioms of Σ_2 inconsistent, we simply add $f_\Lambda(\varphi) = f_\Omega(\varphi)$ to Ω''_1. The resulting f' is a TI (but not TC)

Ω-restriction of f with a consistent ground space and an inconsistent abstract space. $\square$

By having f_Λ behave the same way as f_Ω, the effects of an abstraction are made more local, and properties such as being TI* depend on localised factors. It is not therefore necessary to ask for global properties like that all Ω-restrictions be TI*.

We end by noting that the stronger the abstraction, the greater the chance of generating an inconsistent abstract space. A stronger abstraction will add more theorems to the abstract space, and will therefore increase the likelihood that one of these theorems will introduce inconsistency (that is, there are more formulae with φ's property in the proof of theorem 2). This is something that might have been expected and confirms the intuition that the more details we throw away, the greater the risk of generating inconsistent abstract spaces.

4.4 Preserving Negation

We have shown in theorem 3 that inconsistent abstract spaces are inevitable for NTI abstractions. To obtain a similar result with TI abstractions (theorem 2) we had to guarantee that the meaning of negation was preserved. But what if we do not preserve the meaning of negation? Of course, there are infinitely many ways of not preserving negation. However, as we argue in detail in [GW92], to modify the logic is a very bad idea. Usually it is the theory and not the logic that should be abstracted. Additionally, negation is very critical as it directly links provability to inconsistency. In the rest of this section, we will look at abstractions which do not preserve negation, and which therefore might not give inconsistent abstract spaces.

1. Let's suppose that both the ground and the abstract space have the classical natural deduction inference rules. Negation is kept in the language of the abstract space but the relevant inference rules are forgotten; in other words, the reasoning by absurdity inference rule is dropped. The resulting logic is the Minimal logic [Pra71] and, if f_Λ and f_Ω are the identity functions, then the resulting abstraction is such that any wff mapping into an abstract theorem is a theorem of the ground space (but not vice versa). This is not a TI abstraction.

2. Assume the same logic as above but negation is not kept in the abstract language (all the instances of negation are deleted). In particular $f(\alpha) = f(\neg\alpha) = \alpha$. $\perp$ is kept in the abstract language and, for any inference rule $\delta \in \Delta_1$:

$$f_\Delta(\delta) = f_\Delta(\frac{\alpha_1, ..., \alpha_n}{\alpha_{n+1}}) = \frac{f_\Lambda(\alpha_1), ..., f_\Lambda(\alpha_n)}{f_\Lambda(\alpha_{n+1})}$$

This last property is called Λ/Δ-invariance. Such an abstraction gives an absolutely inconsistent abstract space. In fact, the inference rule

$$\frac{\begin{array}{c}[\alpha]\\ \bot\end{array}}{\neg\alpha}$$

becomes

$$\frac{\begin{array}{c}[\alpha]\\ \bot\end{array}}{\alpha}$$

and everything can be proved.

3. Another interesting example is an abstraction used in GPS [NSS63, NS72] for logic problems. In this Λ/Δ-invariant abstraction, all the logical connectives are forgotten and a wff is mapped into a tree of atomic wffs; for example, $p \wedge (q \vee r)$ abstracts to $(p, (q, r))$. The abstract space is absolutely inconsistent. In fact, by a sequence of (abstractions of) $\vee I$ and $\wedge E$, any wff can be derived.

4. Abstraction has been used to plan the unfolding of definitions. The UT prover [BT75] is a powerful ND theorem prover developed at the University of Texas which uses a local heuristic called *peeking* to control the expansion of definitions. Definitions are only unfolded if they introduce predicate names that are mentioned in the conclusion of the theorem to be proved. This was generalised to *gazing*, a global strategy for controlling the unfolding of definitions [Plu87]. At the heart of gazing is the *common currency model*; namely the idea of finding a common language of concepts between hypotheses and conclusion. After the unfolding the proof can then be completed by logical inference alone. Gazing constructed the plan of definitions to unfold in a hierarchy of abstraction spaces: the predicate space (which we shall consider here) and the function/polarity space (which for lack of space we shall not consider). We can formalise gazing as an abstraction between a first order calculus and a propositional calculus. For any wff which is not a definition, we forget the arguments of the predicates and throw away negation (Gazing actually worked on sets of propositions which were implicitly disjoined in the conclusion and conjoined in the hypotheses). For a definition, we just forget the arguments of the predicates. This is neither a TI nor a NTI abstraction (Note that an abstraction which deletes all arguments from predicate symbols making them into propositional constants is both TI and NTI). It does not give a complete nor a sound strategy for deciding when to unfold definitions; not only does it sometimes fail to suggest the appropriate definitions to unfold, it also can suggest the wrong definitions. See [GW89b] for a longer discussion.

5 Conclusions

The generation of inconsistent abstract spaces is quite a well known problem. The results presented here are, however, more general: we have shown that inconsistent abstract spaces are generated because the set of abstract theorems is a strict superset of the abstraction of the ground theorems. Our results hold for a class of TI*-abstractions that captures practically all the abstractions used in the past. Abstractions throw away details; *what is important is the impact this has on provability.* The examples of inconsistent abstract spaces identified in the past were restricted to particular abstractions and particular types of formal systems (*eg.* Abstrips, Plaisted's abstractions); it was not clear what was the relationship between the different examples, even if it was clear that there must be some relation. We have shown the problem occurs independently of any particular abstraction or formal system.

The problem of inconsistent abstract spaces cannot be avoided a priori if an arbitrary set of axioms is allowed in the ground space. The best that can be done is to tackle the problem at runtime by minimizing the inefficiencies that inconsistent abstract spaces may cause. Thus, when an inconsistent abstract space is generated and we find this out, we should exploit this information. One way is to backtrack and build "more interesting" abstract spaces [Ric83]. Another is to try to discover what led to the inconsistency [Doy86].

References

[BT75] W.W. Bledsoe and M. Tyson. The UT interactive prover. Technical report, Mathematics Department, University of Texas, 1975. ATP-17.

[Doy86] R.J. Doyle. Constructing and refining causal explanations from an inconsistent domain theory. In *Proc. Fifth National Conference on Artificial Intelligence*, Philadelphia, PA, 1986. AAAI.

[GG88] F. Giunchiglia and E. Giunchiglia. Building complex derived inference rules: a decider for the class of prenex universal-existential formulas. In *Proc. 7th European Conference on Artificial Intelligence*, 1988. Extended version available as DAI Research Paper 359, Dept. of Artificial Intelligence, Edinburgh.

[Gre69] C. Green. Application of theorem proving to problem solving. In *Proc. 1st IJCAI conference*, pages 219–239. International Joint Conference on Artificial Intelligence, 1969.

[GW89a] F. Giunchiglia and T. Walsh. Abstract Theorem Proving. In *Proc. IJCAI 89*, 1989. IRST Technical Report 8902-03. Also available as DAI Research Paper No 430, University of Edinburgh.

[GW89b] F. Giunchiglia and T. Walsh. Theorem Proving with Definitions. In *Proc. of the 7th Conference of the Society for the Study of Artificial Intelligence and Simulation of Behaviour*, 1989. Also available as IRST Technical Report 8901-03 and DAI Research Paper No 429, Dept. of Artificial Intelligence, Edinburgh.

[GW92] F. Giunchiglia and T. Walsh. A Theory of Abstraction. *To appear in: Artificial Intelligence Journal*, 1992. Also IRST-Technical Report 9001-14, IRST, Trento, Italy.

[Hob85] J.R. Hobbs. Granularity. In *Proc. 9th IJCAI conference*, pages 432–435. International Joint Conference on Artificial Intelligence, 1985.

[Men64] E. Mendelson. *Introduction to Mathematical Logic.* Van Nostrand Reinhold, 1964.

[MH69] J. McCarthy and P. Hayes. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edimburgh University Press, 1969.

[Nil80] N.J. Nilsson. *Principles of Artificial Intelligence.* Tioga Publishing Co., 1980.

[NS72] A. Newell and H.A. Simon. *Human Problem Solving.* Prentice-Hall, 1972.

[NSS63] A. Newell, J.C. Shaw, and H.A. Simon. Empirical explorations of the logic theory machine. In Fiegenbaum and Feldman, editors, *Computers & Thought*, pages 134–152. McGraw-Hill, 1963.

[Pla80] D.A. Plaisted. Abstraction mappings in mechanical theorem proving. In *5th Conference on Automated Deduction*, pages 264–280. Proc. of the 5th Conference on Automated Deduction, 1980.

[Pla81] D.A. Plaisted. Theorem proving with abstraction. *Artificial Intelligence*, 16:47–108, 1981.

[Pla86] D.A. Plaisted. Abstraction using generalization functions. In *8th Conference on Automated Deduction*, pages 365–376. Proc. of the 8th Conference on Automated Deduction, 1986.

[Plu87] D. Plummer. *Gazing: Controlling the Use of Rewrite Rules.* PhD thesis, Dept. of Artificial Intelligence, University of Edinburgh, 1987.

[Pra65] D. Prawitz. *Natural Deduction - A proof theoretical study.* Almquist and Wiksell, Stockholm, 1965.

[Pra71] D. Prawitz. Ideas and results in proof theory. In J.E. Fenstad, editor, *Proc. 2nd scandinavian logic symposium.* North Holland, 1971.

[Ric83] E. Rich. *Artificial Intelligence.* McGrew-Hill, New York, 1983.

[Sac74] E.D. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5:115–135, 1974.

[Ten87] J.D. Tenenberg. Preserving Consistency across Abstraction Mappings. In *Proc. 10th IJCAI conference*, pages 1011–1014. International Joint Conference on Artificial Intelligence, 1987.

[Ten88] J.D. Tenenberg. *Abstraction in Planning.* PhD thesis, Computer Science Department, University of Rochster, 1988. Also TR 250.

[UR89] A. Unruh and P. Rosenbloom. Abstraction in problem solving and learning. In *Proc. 11th IJCAI conference.* International Joint Conference on Artificial Intelligence, 1989.