Irrelevance in Problem Solving

Alon Y. Levy
Knowledge Systems Laboratory
Stanford University
701 Welch Road, Bldg. C, Palo Alto, CA 94304
alevy@cs.stanford.edu

Abstract

The notion of irrelevance underlies many different works in AI, such as detecting redundant facts, creating abstraction hierarchies and reformulation and modeling physical devices. However, in order to design problem solvers that exploit the notion of irrelevance, either by automatically detecting irrelevance or by being given knowledge about irrelevance, a formal treatment of the notion is required.

In this paper we present a general framework for analyzing irrelevance. We discuss several properties of irrelevance and show how they vary in a space of definitions outlined by the framework. We show how irrelevance claims can be used to justify the creation of abstractions thereby suggesting a new view on the work on abstraction.

Introduction

Meta-level reasoning has received a lot of attention from researchers in artificial intelligence as a means of guiding problem solvers in their search for solutions [Hayes, 1973; Genesereth, 1988; Smith and Genesereth, 1985; Clancey, 1983]. A common of meta-level strategy is to avoid using knowledge that is irrelevant to the goal at hand. In fact, the notion of irrelevance has been a common theme in many research works, but its formal analysis has received attention only from few researchers such as Subramanian and Genesereth [Subramanian and Genesereth, 1987; Subramanian, 1989]. The ability to give a problem solver advice about what parts of a knowledge base are irrelevant to a specific problem solving goal is a powerful method to reduce its search. For example, consider a domain in which we are trying to find routes between cities in the country, using flights, trains and busses. For some goals, we might want to advise the problem solver that rules and facts about flights are irrelevant, either because the minimal price of flights is known to be greater than is required for the specific goal or because we know that flights will not yield an optimal solution. By giving this advice, we significantly reduce the size of the search space explored by the problem solver.

The notion of irrelevance also plays a key role in work on abstractions and change of representation. Intuitively, when we want to create a simpler or abstract representation we remove some irrelevant detail. If the removed detail was indeed irrelevant then the solution to the problem in the abstract theory will map back to a solution in the original theory. Therefore, if we can provide the system with knowledge about irrelevance or relative irrelevance of knowledge, the system can exploit it to automatically create abstractions. Methods for mechanically detecting relevance can be used to automatically create abstractions.

However, both in order for a user to be able to state such claims to a system in a principled manner and for the system to make proper use of given claims, a better analysis of the notion of irrelevance in problem solving is required. This paper describes a general framework for analyzing the notion of irrelevance. We define a space of possible definitions of irrelevance by identifying several axes along which irrelevance claims differ. Several important properties of irrelevance concerning their usage in problem-solving are outlined and we show how varying the definition of irrelevance in our space affects the satisfaction of these properties. Next, we discuss how irrelevance claims can serve as justifications for creating an abstraction. The case of irrelevance of a distinction between properties (represented as predicates) is examined in detail and we show how such a claim serves as a justification for predicate abstraction [Plaisted, 1981; Tenenberg, 1987].

This framework makes several contributions. First, it clarifies the issues involved in the notion of irrelevance therefore enabling us to better exploit the notion in works that rely on it, such as the work on detecting redundant facts or creating abstraction hierarchies. The properties of irrelevance that we outline provide guidance in building a system that incorporates such claims. Giving precise definitions of irrelevance formalizes the problem of automatically deducing irrelevance facts, thereby enabling us to automatically create abstractions, based on deduced irrelevance claims. Moreover, since our framework provides a language to express knowledge about irrelevance, we can use this language to express knowledge about the domain that can help reduce the size of the search or justify creating an abstraction.

Preliminaries

Assume our theory of the domain is represented by a knowledge base of first order predicate calculus formulas, Δ. A problem solving goal (or query) is represented by a formula ψ. The goal is to find whether ψ is implied by Δ (or if ψ has free variables, we want to know which assignments to the variables result in a formula that is entailed from Δ.). Our aim is to identify facts that are *irrelevant* to ψ in order to reduce the search space generated for ψ. Formalizing the concept of irrelevance can be done in several levels. For example, one can formalize irrelevance in terms of the models of Δ and ψ, i.e., a semantic level analysis. Irrelevance can also be analyzed in terms of the facts in the theory Δ, a so called *meta-theoretic* analysis [Subramanian, 1989]. Alternatively, one can give a proof-theoretic analysis of irrelevance, in terms of the actual set of derivations the problem solver can explore in the search to solve ψ. Although these levels are by no means independent, it is important to distinguish between them when defining irrelevance or comparing between definitions.

The goal of this paper is to define notions of irrelevance that enable us to optimize actual problem solving. Therefore, we analyze irrelevance from the system's view of the problem-solving process which is a proof-theoretic one. The system does not actually see the world as the user sees it nor does it see the conceptualization of the world. Instead, it sees the set of symbols used to describe the domain and the set of derivations it can generate.

Example 1: Suppose we are using a resolution theorem prover on a knowledge base in clause form[1]. Consider the following two theories:

$T_1 = \{f \Rightarrow g, \neg f \Rightarrow g\}$

$T_2 = \{g\}$.

T_1 and T_2 are satisfied by the same set of models. In each the value assigned to f does not affect the value of g, and therefore we might consider f to be irrelevant to g. However, in T_1, the theorem prover will have to resolve on the symbol f to derive g, and therefore as far as it is concerned, it can't ignore the symbol f. ▮

Note that we are not claiming that irrelevance relations in the domain are not useful to control problem solving; quite the contrary. Most irrelevance facts are based on properties of the domain. However, a relevance relation in the domain will only be useful if it is reflected in the representation.

In particular, for a problem solver to exploit irrelevance claims, the following properties of irrelevance claims will be of interest. Assume $IR(\phi, \psi, \Delta)$ denotes that the fact (or set of facts) ϕ is irrelevant to the goal ψ with respect to the theory Δ.

- What can the problem solver do given the irrelevance claim? Can it ignore a fact that is deemed irrelevant? Can it ignore any fact that contains it as a subexpression?
- Do irrelevance claims add up? If $IR(\phi_1, \psi, \Delta)$ and $IR(\phi_2, \psi, \Delta)$ hold, does that imply that $IR(\{\phi_1, \phi_2\}, \psi, \Delta)$ holds? If so, we can use all the relevance claims that are available to us at a given instant. However, if not, we can only use one at a time, and then we must check that the others still hold in the resulting theory.
- Is irrelevance a monotonic property? I.e., if we add more facts to the knowledge base, can irrelevant facts become relevant or vice versa?
- Does the irrelevance of a subject imply the irrelevance of a subject which is syntactically related to it? E.g., Does $IR(\phi, \psi, \Delta)$ imply $IR(\neg\phi, \psi, \Delta)$ or $IR(\phi \vee \phi_1, \psi, \Delta)$? Such properties will enable us use a given set of irrelevance claims to deduce additional ones.
- Can irrelevance claims be found automatically by examining the KB?

An important issue in a definition of irrelevance is the *subject* of irrelevance, i.e., the type of entity being deemed irrelevant to the goal. So far we discussed only the irrelevance of a fact (or set of facts) to a problem solving goal, but the subject may be any kind of entity in the representation, such as the objects-constants, predicate-symbols and functions. The irrelevance subject can also be more abstract such as a decision to distinguish between a set of predicates or objects in the representation. The following is an example of the irrelevance of a predicate distinction.

Example 2: Consider the knowledge base with the following facts.

$r_1 : SportsCar(x) \Rightarrow Vehicle(x)$
$r_2 : FamilyCar(x) \Rightarrow Vehicle(x)$
$r_3 : SportsCar(x) \Rightarrow HighRiskInsurance(x)$
$r_4 : FamilyCar(x) \Rightarrow \neg SportsCar(x)$
$r_5 : FamilyCar(Camry)$

In order to solve the query $Vehicle(x)$, the distinction between the relations $SportsCar$ and $FamilyCar$ is irrelevant. Intuitively, all that matters for the proof is that x is some kind of car. Therefore, we can remove the distinction in the representation by *predicate abstraction* [Tenenberg, 1987]. We express the theory using an abstract predicate, Car, as follows:

[1]For clarity, in this document we do not use clause form notation but assume the problem solver gets formulas in clause form.

$s_1 : Car(x) \Rightarrow Vehicle(x)$
$s_2 : Car(Camry)$

r_1 and r_2 were abstracted to s_1, while r_5 was abstracted to s_2. r_3 on the other hand was a rule specific to *SportsCar*, because
$FamilyCar(x) \Rightarrow HighRiskInsurance(x)$
does not hold, and therefore we cannot abstract it to $Car(x) \Rightarrow HighRiskInsurance(x)$.
Consequently, it is removed from the theory. Similarly, r_4 is a formula that distinguishes between the relations *FamilyCar* and *SportsCar* and therefore is removed from a theory that ignores the distinction between these relations. ∎

A final issue that factors into a definition of irrelevance is the space of possible changes of the representations and the theory (or *weakenings* of the theory [Subramanian and Genesereth, 1987]) we are considering in order to remove the irrelevancy. In example 1, we only considered changing the theory by removing facts and therefore we could not justifiably say that f is irrelevant to g. However, had we considered changing the theory by adding some of its logical consequences (e.g., g), we could deem f irrelevant to g. In example 2, the irrelevancy was removed by predicate abstraction, i.e., replacing the predicates *FamilyCar* and *SportsCar* by an abstract predicate *Car*.

A Space of Irrelevancies

To capture the various properties of irrelevance we define a space of possible definitions of irrelevance. The space of definitions revolves around the set of possible derivations of the goal. Let Δ be a knowledge-base, ψ be a goal and $\mathcal{D}$ be the set of derivations of ψ from Δ. A definition of irrelevance of ϕ (which can be any irrelevance subject) to ψ is composed of the following choices:

A1. Defining irrelevance of ϕ with respect to a single derivation, $D \in \mathcal{D}$.

A2. A subset $\mathcal{D}_0$ of $\mathcal{D}$ over which to quantify A1.

A3. The method of quantification over $\mathcal{D}_0$, i.e., existentially or universally.

Formally, if D is a derivation of a goal ψ from a knowledge base, Δ, we denote the choice for A1 by $Ir(\phi, \psi, D)$, i.e., that ϕ is irrelevant to the derivation D of the goal ψ. If Φ is a set of facts, $Ir(\Phi, \psi, D)$ holds if $Ir(\phi_i, \psi, D)$ for all $\phi_i \in \Phi$.

Definition 3: Let $\mathcal{D}_0$ be a set of derivations of a goal ψ from the knowledge base Δ[2]. ϕ is said to be ***weakly irrelevant*** to ψ with respect to $\mathcal{D}_0$, denoted by $WI(\phi, \psi, \mathcal{D}_0)$[3], if $Ir(\phi, \psi, D)$ holds for some $D \in \mathcal{D}_0$. ϕ is said to be ***strongly irrelevant***, denoted by $SI(\phi, \psi, \mathcal{D}_0)$, if $Ir(\phi, \psi, D)$ holds for every $D \in \mathcal{D}_0$. ∎

[2]If ψ is a set of goals (e.g., a goal with free variables) we consider a set of derivations for every element of ψ. The definitions below hold if they hold for every element of ψ.

[3]Note that the knowledge base Δ is implicit in the third argument of WI and SI.

Note that in Definition 3, the knowledge base, Δ does not appear explicitly in WI (SI), but is implicit in the set $\mathcal{D}_0$. For every choice of Ir and of $\mathcal{D}_0$, we get a definition for weak and strong irrelevance. Except for $\mathcal{D}_0 = \mathcal{D}$, examples of $\mathcal{D}_0$ include the set of all minimal derivations[4], or all derivations bounded by some resource constraints. The following example clarifies some of these distinctions.

Example 4: Consider a knowledge base with the following rules:
$r_1 : E(x) \Rightarrow Q(x)$
$r_2 : R(x) \Rightarrow Q(x)$
$r_3 : P(x) \Rightarrow Q(x)$
$r_4 : E(x) \Rightarrow P(x)$
$r_5 : Q(x) \Rightarrow P(x)$

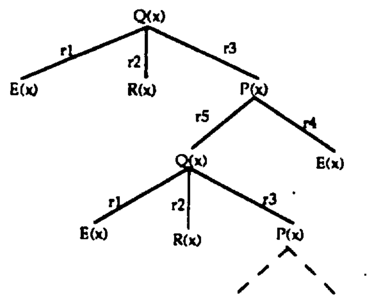

Figure 1: Search space for a goal $Q(x)$

The knowledge base also contains a set of ground facts but only for the predicate E. Figure 1 shows the possible derivations that can be generated for Q from this theory. Suppose we define $Ir(r, g, D)$ to hold if the rule r does not appear in the derivation D. Let $\mathcal{D}$ be the set of all derivations of $Q(a)$[5]. $WI(r_3, Q(a), \mathcal{D})$ holds since whenever $Q(a)$ is derivable, there will be a derivation of $Q(a)$ using only r_1. $SI(r_2, Q(a), \mathcal{D})$ holds because r_2 cannot be part of a proof of $Q(a)$. $SI(r_5, Q(a), \mathcal{D})$ does not hold, however, if we consider the set of non-redundant derivations $\mathcal{D}_0$[6], then $SI(r_5, Q(a), \mathcal{D}_0)$ holds. ∎

Irrelevance of a Fact

In this section we briefly consider the case in which the relevance subject is a single fact, and show how varying the choices for A1-A3 affects the properties of the resulting irrelevance claims. The definitions consider a specific problem solver, hence our discussion assumes we are using resolution theorem prover. A derivation

[4]Given some criteria of minimality of deductions.

[5]which will be empty if $E(a)$ is not in the knowledge base.

[6]A derivation tree is redundant if it has two identical nodes n_1 and n_2 such that n_1 is an ancestor of n_2.

is a resolution tree of clauses, where the goal clause (or the empty clause in case of a refutation proof) is the root, and the children of every clause are the two clauses that were resolved in order to get it. The leaves of the tree are clauses from the knowledge base, and they are denoted by $Base(D)$.

Consider three choices for A1. In the first, a fact is irrelevant to a derivation of the goal if it is not one knowledge-base facts used in it:

Definition 5: $Ir_1(\phi, \psi, D)$ if $\phi \notin Base(D)$. ▮

A stronger definition requires that ϕ is irrelevant to a derivation if it appears nowhere in the derivation:

Definition 6: $Ir_2(\phi, \psi, D)$ iff there does not exist a substitution σ such that $\phi\sigma$ is a subclause of a clause in D. ▮

Subramanian [Subramanian, 1989] defines ϕ to be irrelevant to ψ with respect to a theory Δ, if there is a subset of Δ that ils ψ but is non-committal on ϕ. In our space, we n formalize this as follows:

Definition 7: $Ir_3(\phi, \psi, D)$ if $Base(D) \not\models \phi$ and $Base(D) \not\models \neg\phi$. ▮

Using Ir_3, for a refutation resolution theorem prover, $WI(\phi, \psi, \mathcal{D})$ is equivalent to the definition given in [Subramanian, 1989].

Figure summarizes the different properties that hold for the definitions described above. The following show how the properties of weak irrelevance differ from those of strong irrelevance.

Observation 8: Whenever irrelevance adds up on a single derivation, it will add up for strong irrelevance, i.e., if

$$Ir(\Phi_1, \psi, D) \wedge Ir(\Phi_2, \psi, D) \Rightarrow Ir(\{\Phi_1, \Phi_2\}, g, D)$$

hold for any D, then for any choice of $\mathcal{D}_0$,

$$SI(\Phi_1, \psi, \mathcal{D}) \wedge SI(\Phi_2, \psi, \mathcal{D}) \Rightarrow SI(\{\Phi_1, \Phi_2\}, g, \mathcal{D})$$

This property does not hold for weak irrelevance. ▮

Observation 9: The converse holds for weak irrelevance too, i.e., whenever

$$Ir(\{\Phi_1, \Phi_2\}, \psi, D) \Rightarrow Ir(\Phi_1, \psi, D) \wedge Ir(\Phi_2, \psi, D)$$

holds for any D, then for any choice of $\mathcal{D}_0$,

$$WI(\{\Phi_1, \Phi_2\}, \psi, \mathcal{D}_0) \Rightarrow WI(\Phi_1, \psi, \mathcal{D}_0) \wedge WI(\Phi_2, \psi, \mathcal{D}_0)$$

$$SI(\{\Phi_1, \Phi_2\}, \psi, \mathcal{D}_0) \Rightarrow SI(\Phi_1, \psi, \mathcal{D}_0) \wedge SI(\Phi_2, \psi, \mathcal{D}_0)$$

▮

Observation 10: For any definition of Ir such that $Ir(\phi, \psi, D) \Rightarrow Ir_1(\phi, \psi, D)$, if we add facts to the knowledge base, irrelevance can change as follows. A fact that was weakly irrelevant will still be weakly irrelevant. A fact that was strongly irrelevant will be at least weakly irrelevant. A fact that was not weakly irrelevant might become weakly irrelevant. ▮

	Ir_1	Ir_2	Ir_3
P_1	✓	✓	✓
P_2		✓	
P_3		✓	
P_4	✓	✓	✓
P_5			✓
P_6		✓	

P_1: $WI(\phi, \psi, \mathcal{D})$ implies that $\Delta \setminus \phi \vdash \psi$.
P_2: $WI(\phi, \psi, \mathcal{D})$ implies that the problem solver can ignore any derivation that contains ψ.
P_3: $WI(\phi, \psi, \mathcal{D})$ implies that the problem solver can ignore any derivation that contains ψ as a subexpression.
P_4: *Adding up* -
$Ir(\Phi_1, \psi, D) \wedge Ir(\Phi_2, \psi, D) \Rightarrow Ir(\{\Phi_1, \Phi_2\}, g, D)$.
P_5: *Transfers through equivalence* -
$Ir(\phi_1, \psi, D) \wedge (\phi_1 \equiv \phi_2) \Rightarrow Ir(\phi_2, \psi, D)$.
P_6: If ϕ is a subclause of ϕ_1, then
$Ir(\phi_1, \psi, D) \Rightarrow Ir(\phi_1, \psi, D)$.

Figure 2: Properties of Irrelevance

Deducing Irrelevance Claims

Varying the definition of irrelevance has drastic effects on the ability to automatically derive irrelevance claims. Given a knowledge base Δ and a goal ψ, we would like to derive all (or part of) the facts in Δ that are irrelevant to ψ. In general, looking at the whole knowledge base to determine irrelevance will be more costly than solving the query. A more interesting question is whether irrelevance claims can be derived by looking at only a small and stable part of the knowledge base. For example, in example 4, we were able to determine irrelevance by merely looking at the structure of the proof space created by the rules, regardless of the specific ground facts for the predicate E.

We examine this question for knowledge bases comprised of a set of Horn rules with no function symbols (Datalog, [Ullman, 1989]), and a database of ground atomic facts. We distinguish between two sets of predicates in the knowledge base, the *extensional predicates* (EDB predicates) which are those that appear only in the database and in antecedents of rules, and the *intensional predicates* (IDB predicates) which are the predicates appearing in the consequents of the rules, i.e., the predicates that are being defined by the EDB predicates and the rules. A query is an IDB predicate, i.e., to find all the derivable facts for that predicate. Every derivable instance of the goal has a (perhaps more than one) derivation tree. A *derivation tree* is a tree consisting of *goal-nodes* and *rule-nodes*. A goal node is labeled by a ground atom, and it has a single child, which is an instantiated rule-node. The head of an instantiated rule-node is identical to its parent goal-node. A rule-node has a child goal-node for each one of its subgoals. The leaves of a derivation tree are goal-nodes labeled by ground atoms from the EDB. A derivation is *not minimal* (or *redundant*) if there are two identical goal-nodes n_1 and n_2, such n_1 is an ancestor of n_2. A rule r is irrelevant to a derivation D (i.e., $Ir(r, \psi, D)$) if none of the rule nodes in D are instances of r (note that this is equivalent to Ir_1 and Ir_2).

The question we address is the following. Given a set of rules, $\mathcal{P}$, a query q and a definition of irrelevance, can we determine whether a rule $r \in \mathcal{P}$ is irrelevant to query for *any* possible set of ground facts in the knowledge base. We consider two choices for A2, the set of all derivations of the goal q, denoted by $\mathcal{D}$, and the set of all minimal derivations, $\mathcal{D}_0$.

Finding irrelevant rules enables us to significantly prune the search space for the query. In example 4, rule r_2 will not appear in any derivation of Q, therefore $SI(r_2, Q(x), \mathcal{D})$ holds. r_5 will appear only in redundant derivations of Q and therefore $SI(r_5, Q(x), \mathcal{D}_0)$ holds. Since $Q(x)$ can always be derived using either r_1 or $\{r_3, r_4\}$, both $WI(r_1, Q(x), \mathcal{D})$ and $WI(\{r_3, r_4\}, Q(x), \mathcal{D})$ hold. Consequently, identifying the various kinds of irrelevance can enable us to compute Q using only r_1. Considering constraint literals in the rules enables us to derive additional irrelevance claims:

Example 11: Consider the following knowledge base:

$s_1 : Q(x, z) \wedge Q_1(z, y) \wedge x < z \Rightarrow P(x, y)$
$s_2 : Q(z, x) \wedge Q_1(x, y) \wedge x < y \Rightarrow P(x, y)$
$s_3 : E_1(x, y) \wedge x < 3 \Rightarrow Q(x, y)$
$s_4 : E_2(x, y) \wedge x > 1 \Rightarrow Q_1(x, y)$

If the query is $P(x, y)$, all rules are relevant. However, if the query is $P(x, y) \wedge (y < 1)$, then s_2 is strongly irrelevant, i.e., $SI(s_2, P(x, y) \wedge (y < 1), \mathcal{D})$. ▮

Finding all rules which are weakly irrelevant, i.e., $WI(r, g, \mathcal{D})$, is precisely the rule redundancy problem shown to be undecidable by Shmueli [Shmueli, 1987]. Consequently, determining $WI(r, g, \mathcal{D}_0)$ is also undecidable. For strong irrelevance, if the rules contain no constraint literals and no object constants, determining $SI(r, g, \mathcal{D}_0)$ is equivalent to the rule reachability problem that has an easy polynomial time solution [Kifer, 1988]. [Levy and Sagiv, 1992] gives an algorithm for detecting $SI(r, g, \mathcal{D}_0)$ and $SI(r, g, \mathcal{D})$ even when constraint literals are present. It also establishes an exponential-time lower bound on the problem of determining $SI(r, g, \mathcal{D}_0)$.

Using Irrelevance to Justify Abstractions

Much of the work in AI on creating abstraction hierarchies relies on the intuition that by creating an abstract theory we are removing some irrelevant detail. If the detail removed is indeed irrelevant, then a solution to the problem in the abstract theory will map back to a solution in the original theory (also referred to as the *ground theory*). Otherwise, we will have to backtrack between abstraction levels. Although this has been the motivation underlying work on abstractions, the formal connection between irrelevance and abstractions has received little attention (e.g., [Subramanian, 1989]). For example we can view *predicate abstraction* as being justified by the irrelevance of a distinction between predicates; object aggregation can be justified by irrelevance of a granularity distinction. Identifying abstraction with the notion of irrelevance offers several advantages:

- We make explicit what is being abstracted (i.e., the subject of irrelevance).
- We make clear the strength of the justification for the abstraction (by the strength of the type of irrelevance claim that holds).
- We formalize the problem of automatically creating abstractions by translating it to the problem of automatically finding irrelevance claims.

In this section we briefly discuss how irrelevance claims that are justifications for abstractions can be formulated in our framework. We identify several *irrelevance subjects* that account for many abstractions discussed in the literature. As a consequence we get an expressive language to state knowledge about the domain that can affect the creation of abstractions. We define a notion of irrelevance that best justifies abstractions and mention several weaker notions.

The first assumption underlying a formalization of irrelevance is that removing irrelevant detail should not enable us to reach new conclusions about the set of goals we are interested in, i.e., any conclusion reached in the abstract theory should be an abstraction of one in the base theory (this is also known as a TD property of abstractions [Giunchiglia and Walsh, 1991] or the downward solution property [Tenenberg, 1987]). The justification for this claim is that by removing irrelevant detail, we are effectively ignoring some of our knowledge, and therefore, we can not come to new conclusions[7]. For example, when we remove some irrelevant detail in a planning problem (e.g., action precondition), if the resulting abstract plan can not be mapped back to a base-level plan, the detail we have removed was not truly irrelevant to the problem[8]. Sec-

[7]As long as the our reasoning has no form of non-monotonicity.

[8]Note that this does not necessarily mean that the abstraction is not useful!

ond, the abstract theory should not prevent us from solving the goal, i.e., if the original theory had a solution to the goal, then the abstract one should too. Finally, in order for the abstraction to be computationally effective, the solutions that are preserved by the abstraction should be the cheaper ones.

These criteria are naturally formulated in our framework. Recall that in order to define irrelevance of a subject α, we must give a definition for $Ir(\alpha, \psi, D)$, i.e., when the subject α is irrelevant to a derivation D. Given a theory Δ, we denote the abstract theory resulting from removing the irrelevancy α by $f_\alpha(\Delta)$. For example, if α is a distinction between predicates, $f_\alpha(\Delta)$ is the theory resulting from predicate abstraction. The exact form of $f_\alpha(\Delta)$ is discussed in the next section. We base our definition of Ir on a mapping h_α from the derivations of ψ in Δ, denoted by $\mathcal{D}_1$, to the derivations of $f_\alpha(\psi)$ in $f_\alpha(\Delta)$, $\mathcal{D}_2$. The only requirement from h_α is that it is onto $\mathcal{D}_2$. h_α need not be a total mapping on $\mathcal{D}_1$, i.e., there might be derivations of ψ that will not be mapped to the abstract theory, and it need not be 1-1. Other constraints on h_α will yield stronger forms of irrelevance and therefore stronger justifications for the abstraction, (for example, h_α will be called a *simplifying mapping* if for any $D \in \mathcal{D}_1$, the cost of $h(D)$ is no more than the cost of D[9]). Given h_α, $Ir(\alpha, \psi, D)$ is defined as follows:

Definition 12: $Ir(\alpha, \psi, D)$ is true iff $h_\alpha(D)$ is not empty. ▮

Note that in this definition h_α is dependent on α and ψ. Definitions of weak and strong irrelevance are obtained by quantifying the definition of Ir over a chosen set of derivations. The following states that the first two requirements of an abstraction are satisfied by weak irrelevance.

Observation 13: If $\mathcal{D}_0$ is a set of derivations in $\mathcal{D}_1$, and $WI(\alpha, \psi, \mathcal{D}_0)$ holds then ψ is provable from Δ if and only if $f(\psi)$ is provable from $f_\alpha(\Delta)$. ▮

In order satisfy the third requirement, we must impose a restriction on $\mathcal{D}_0$:

Observation 14: If $\mathcal{D}_0$ is a set of derivations that contains all minimal derivations and h_α is a simplifying mapping, then if $SI(\alpha, \psi, \mathcal{D}_0)$ holds, $f_\alpha(\psi)$ will have a solution in the abstract theory if and only if it has one in the original theory, and at least one of abstract-level solutions will cost no more than that cheapest solution in the original theory. ▮

This condition is a sound justification for creating the abstraction. Imposing more constraints on h_α will give us even stronger justifications. For example, we can require that $h_\alpha(D)$ effectively break up D into subproblems of equal size. Knoblock [Knoblock, 1990; Knoblock *et al.*, 1991] shows how this constraint along with other affects the ability to achieve savings when employing hierarchical planning.

Weaker irrelevance claims can also be given to the system. For example, we can state a distinction α_1 is more relevant than a distinction α_2, i.e., whenever α_1 is justifiably abstracted, so is α_2. Another kind of claim is one a probabilistic one, i.e., stating to the system that in most cases α is irrelevant to ψ. The system can then use this claim and succeed in most cases and backtrack in others. By stating irrelevance claims declaratively we can also state under what conditions the relevance claim holds.

In the next section we examine the case of predicate abstractions and show they are justified by irrelevance of a predicate distinction.

Irrelevance of Predicate Distinctions

When designing a representation, a decision has to be made about the detail with which to conceptualize the world. In some cases, identifying a property P (e.g., $Car(x)$) will suffice. In other cases we need to refine P to subclasses $\mathcal{P} = \{P_1, \ldots, P_n\}$ (e.g., $SportsCar(x)$, $FamilyCar(x)$, etc.) For some goals, the finer distinction of properties is irrelevant, and therefore, reasoning will be more efficient if we change the theory by abstracting the distinction. We would like to be able to give the system knowledge about the domain that will guide it in deciding when a predicate distinction is relevant. To define the meaning of such an irrelevance claim in the framework, we first must define the abstract theory resulting from removing the predicate distinction and the mapping of derivations between the original and abstract theories.

The Abstract Theory

Suppose we have a theory Δ, consisting of a set of predicates $\mathcal{P} = \{P_1, \ldots, P_n\}$, and we want to abstract the distinction between them by replacing them by a predicate P that represents their union (e.g., we want to replace $\{FamilyCar, SportsCar\}$ by the predicate Car). Intuitively, to abstract the theory Δ, we replace every occurrence of a predicate in $\mathcal{P}$ in every formula in Δ by P (e.g., abstract $FamilyCar(x) \Rightarrow Vehicle(x)$ by $Car(x) \Rightarrow Vehicle(x)$). However, doing so for every formula in Δ might result in an inconsistent theory or in a theory that will entail conclusions that were not entailed by the original one. In example 2, abstracting rule r_4 will result in a contradiction $(Car(x) \Rightarrow \neg Car(x))$, and abstracting r_3 will result in a fact that is not entailed by the theory (i.e., $Car(x) \Rightarrow HighRiskInsurance(x)$ does not follow from the theory). In order to assure that our derivation mapping will be onto, we need the abstract theory to be consistent with the ground one. Tenenberg [Tenenberg, 1987; Tenenberg, [illegible]] discusses predicate abstractions and defines the maximal set of formulas that can be included in the abstract theory such that the abstract theory will be consistent with the original one. His

[9]Given some cost model for derivations such as the number of nodes in the proof tree.

definition is based on the interpretation of the abstract predicate, which is the union of the interpretations of the predicates in $\mathcal{P}$. However, as Tenenberg notes, this set is usually infinite even when the ground theory is finite. Therefore, the abstract theory we consider is a finite subset of the one defined by Tenenberg. Our abstract theory consists of the abstractions of the formulas in the base theory that are *independent* of the predicate distinction. Intuitively, a formula is independent if its abstraction is consistent with the theory. In the formal definition, we assume that formulas are represented as clauses. A literal in a clause is negative if it is a negation of an atomic formula (e.g., $\neg P(x)$ is a negative literal, while $P(x,y)$ is a positive literal). $Neg(C)$ ($Pos(C)$) denotes the set of negative (positive) literals in a clause C.

Definition 15 : Independence - Let $\mathcal{P} = P_1, \ldots, P_n$, and suppose $Neg(C)'$ is the result of substituting every occurrence of an element of $\mathcal{P}$ in $Neg(C)$ by some other predicate in $\mathcal{P}$ using a mapping f_1. (Two occurrences of the same predicate need not have the same mapping under f_1.) A clause C is independent of a predicate distinction $\mathcal{P}$ with respect to a ground theory Δ, if for any such f_1 there exists a mapping, f_2 of the occurrences of elements of $\mathcal{P}$ in $Pos(C)$ to elements of $\mathcal{P}$, such that $Pos(C)' = f_2(Pos(C))$ and $\Delta \vdash Pos(C)' \cup Neg(C)'$[10] ▮

Note, that a clause that contains only positive literals from $\mathcal{P}$ will be independent whenever it is provable from the theory. The problem arises with the negative literals. In example 2, all rules but r_3 are independent of the distinction $\{FamilyCar, SportsCar\}$.

Lemma 16: *A clause C is independent of a predicate distinction $\mathcal{P}$, if and only if $f(C)$ would be included in the abstract theory as defined by Tenenberg in [Tenenberg, 1990].*

The Derivation Mapping

Given the abstract theory produced by removing the predicate distinction, we can define the mapping of derivations in the base-theory to those in the abstract one. Recall that we require that the mapping be an onto mapping. Intuitively, given a derivation in the abstract theory, a base-level derivation that is mapped to it should be obtainable by reversing the abstraction function on the formulas in the derivation. However, as the following example shows, this cannot always be done.

Example 17: Consider the following knowledge base:
$r_1 : P_1(x) \Rightarrow Q(x)$
$r_2 : P_2(x) \Rightarrow R(x)$
$r_3 : R(x) \Rightarrow P_1(x)$
$r_4 : P_2(a)$

[10]Notice, that in the definition we use $\vdash$, which assumes a simple case where the base-level reasoner and the meta-level reasoner are the same. However, in general, they need not be the same.

Suppose we want to abstract P_1, P_2 by an abstract predicate P. The resulting abstract theory will be:
$s_1 : P(x) \Rightarrow Q(x)$
$s_2 : R(x) \Rightarrow P(x)$
$s_3 : P(a)$

s_1 is included in the abstract theory because r_1 is independent of the predicate distinction (because $P_2(x) \Rightarrow Q(x)$ is derivable from the theory).

The (single) derivation of $Q(a)$ in the abstract theory cannot be trivially mapped to a base-level derivation. The reason is that it uses s_1 and s_3, and they are abstractions of of r_1 and r_4 which do not yield a base level derivation of $Q(a)$. ▮

The source of the problem is that some reasoning was done in the process of creating the abstract theory. In this case, s_1 already represented a base-level chain of reasoning that derived $P_2(x) \Rightarrow Q(x)$.

Informally, we define the derivation mapping, h_α, by specifying all the base-level derivations that map to a given abstract-level derivation D. The mapping is defined in two steps as follows. Given D, we first construct all the possible mappings in which occurrences of P in D are mapped to elements of $\mathcal{P}$, such that the resulting derivation is a valid one. For example, in Figure 3, the abstract-level derivation (a) has two such possible mappings (b) and (c). Next try to complete each of the resulting derivations such that they will be valid derivation in the base-level theory. In our example, (b) cannot be completed because $P_1(a)$ does not follow from our original theory. (c) however, can be completed, as shown in (d). Any such complete base-level derivation is mapped to D under the mapping h_α. In Figure 3 only (d) is mapped to the abstract level derivation (a) (i.e., $h_\alpha(d) = a$).

In order to show that h_α is onto, we must show that at least one of the intermediate derivations can be completed to a valid derivation from the base-level theory.

We prove this by defining one mapping $\mathcal{M}$, from the occurrences of P in D to $\mathcal{P}$. $\mathcal{M}$ will have the property that when we apply it to D, the resulting derivation is guaranteed to have a completion to a valid base-level derivation. Let $\mathcal{C}$ be the leaves of the abstract level derivation, D that contain the predicate P. We define $\mathcal{M}$ on the occurrences of P in $\mathcal{C}$ such that two literals that are resolved somewhere in D are assigned the same predicate in $\mathcal{P}$. That ensures that $\mathcal{M}$ can be extended to all the occurrences of P in D. For clarity, we assume that P does not appear in the root of D, and that D did not have any non-trivial factoring (see [Genesereth and Nilsson, 1987]). We define a partial order $<$ on the clauses in $\mathcal{C}$, and make assignments to clauses in the topological order induced by $<$.

Definition 18: For every $C_i, C_j \in \mathcal{C}$, $C_i < C_j$ iff an ancestor of C_i is resolved with an ancestor of C_j on a literal in $\mathcal{P}$, and the ancestor of C_i contributes the positive literal to the resolution. ▮

Lemma 19: *The relation $<$ is acyclic.*

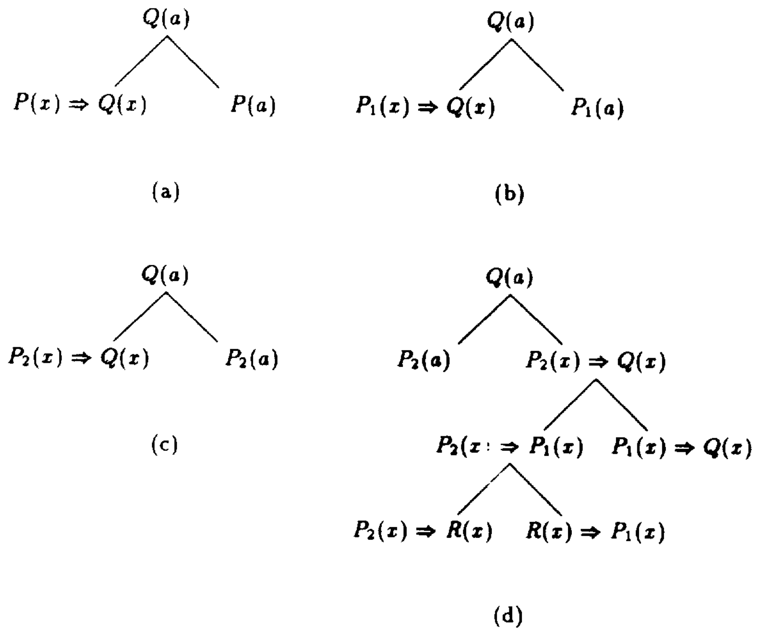
$Q(a)$
$P(x) \Rightarrow Q(x)$
$P(a)$
(a)
$Q(a)$
$P_1(x) \Rightarrow Q(x)$
$P_1(a)$
(b)
$Q(a)$
$P_2(x) \Rightarrow Q(x)$
$P_2(a)$
(c)
$Q(a)$
$P_2(a)$
$P_2(x) \Rightarrow Q(x)$
$P_2(x) \Rightarrow P_1(x)$
$P_1(x) \Rightarrow Q(x)$
$P_2(x) \Rightarrow R(x)$
$R(x) \Rightarrow P_1(x)$
(d)

Figure 3: Mapping base-level derivations to abstract-level derivations

Also note that if C_i is minimal in the order $<$, (i.e., there is no C_j such that $C_j < C_i$), then C_i contains only positive appearances of P.

We define $\mathcal{M}$ on the occurrences of P in C_i only after we have defined the mapping for all its occurrences in clauses C_j such that $C_j < C_i$, as follows:

- If C_i contains only positive appearances of P, we map the occurrences of P such that that the resulting clause is entailed from the base-level theory. Note that by the definition of the abstract theory, there must be at least one such mapping for C_i.
- If C_i contains negative literals of P, we do the following. For any negative occurrence of P, the positive literal with which it is resolved in D has been already mapped previously (by the definition of $<$). Hence we map it to the same element of $\mathcal{P}$ to which its counterpart was mapped. As for the positive literals, any assignment for them such that the resulting clause is derivable from the base theory is a valid assignment. The definition of the abstract theory (i.e., all elements of $\mathcal{C}$ are abstractions of independent base-level clauses), guarantees that at least one such assignment exists.

The mapping $\mathcal{M}$ guarantees that every leaf of the tree is either in the knowledge base or is derivable from it. Therefore, the resulting tree can be completed to a full base-level derivation.

Theorem 20: *The derivation mapping h_α is well defined and onto (i.e., every derivation in the abstract theory has at least one derivation in the base theory that maps to it), and is a simplifying abstraction.*

Properties of the Irrelevance Definition

Given the definition of irrelevance, the question arises whether given the original theory and the abstract one, it is possible to decide if the predicate distinction is irrelevant to the goal. The following provides a first step in that direction by identifying a class of derivations that are preserved by the abstraction.

Theorem 21: *If $\mathcal{D}_0$ is a set of derivations of the goal such that for any $D \in \mathcal{D}_0$, all the facts in $Base(D)$ are independent of the predicate distinction $\mathcal{P}$, then $SI(\mathcal{P}, \psi, \mathcal{D}_0)$ holds.*

Observation 22: The converse does not hold. I.e., ψ can have a derivation in the abstract theory, but not have one in the base theory only from independent facts. Example 17 illustrates that.[11] ∎

[11] Note that if we change the definition of independence to require $Pos(C)' \cup Neg(C)' \in \Delta$ instead of $\Delta \vdash Pos(C)' \cup$

From this condition and the algorithms described in [Levy and Sagiv, 1992] we can construct an algorithm for detecting irrelevance of predicate distinctions in the following case:

Corollary 23: *Given a Datalog theory, Δ and $f_{\mathcal{P}}(\Delta)$ which is the abstract theory resulting from removing the distinction between predicates in $\mathcal{P}$, there is an algorithm to determine whether $SI(\mathcal{P}, \psi, \mathcal{D}_0)$ holds for any given set of ground facts, where $\mathcal{D}_0$ is the set of all non-redundant derivations of ψ from Δ.*

Note that creating the abstract theory is in general undecidable because it entails solving the rule redundancy problem. Methods for detecting some classes of redundant rules (e.g., [Sagiv, 1988]) can be used to construct a subset of the theory.

Other Relevance Subjects

The same technique described above can be used to define irrelevance of other kinds of relevance subjects. [Levy, 1992] discusses the following subjects:

- **Object aggregations:** We replace a set of object constants by an aggregate object. E.g., replace the subparts of a component by one object representing the component. For example, in the Missionaries and Cannibals problem [Amarel, 1981], we can replace the sets of missionaries and cannibals by objects denoting their sets.
- **Object distinction:** We replace a set of object constants by a representative object that has only the properties common to all elements of the set (i.e., we replace a set $\mathcal{O} = \{o_1, \ldots, o_n\}$ by an object o, such that $P(o)$ holds iff $P(o_i)$ holds for every $o_i \in \mathcal{O}$. For example, when reasoning about chemical reactions, it is enough to consider only one representative molecule of every type in the chemical formula and that suffices to describe the complete reaction between the substances.
- **Predicate representative:** We replace a set of predicates $\mathcal{P}$ by an abstract predicate that represents their *intersection*.
- **Macro rule:** We replace a set of facts S by a logical consequence, s of S.

Conclusions

We presented a general formal framework for analyzing the notion of irrelevance. The framework contains a space of possible definitions of irrelevance claims that enabled to formalize previous definitions (e.g., [Subramanian, 1989]) and present new ones. We identified several important properties of irrelevance claims and demonstrated how these properties change as we move in the space of definitions. The framework enabled us to irrelevance claims that serve as justifications for abstractions, thereby providing a new view on work in abstractions. Justifying abstractions by irrelevance claims provides a ***first principles*** [Subramanian, 1989] account of abstractions, elucidating questions such as automatically creating abstractions, creating abstractions that are specific for a given goal and using domain knowledge to guide the creation of abstractions. This paper presents only initial work on in this direction and much remains to be explored.

$Neg(C)'$, we will get the converse direction too, i.e, if a goal has a proof in the abstract theory, it will have one in the ground theory in which all facts are independent of the predicate distinction.

Acknowledgements

I would like to thank several people for discussions on the topics discussed in this paper: Adnan Darwiche, Ed Feigenbaum, Richard Fikes, Fausto Giunchiglia, Pat Hayes, Yumi Iwasaki, Hiroshi Motoda, Pandu Nayak and Shuky Sagiv. Much of this work was done while I was visiting the Hitachi Advanced Research Laboratory, and I would like to thank Hitachi, and especially Hiroshi Motoda for their generous support.

References

Amarel, Saul 1981. On representations of problems of reasoning about actions. In Webber, Bonnie L. and Nilsson, Nils J., editors 1981, *Readings in Artificial Intelligence*. Morgan Kaufmann, Los Altos, CA.

Clancey, William J. 1983. The advantages of abstract control knowledge in expert system design. In *Proceedings of the Third National Conference on Artificial Intelligence*, Los Altos, CA. Morgan Kaufmann. 74–78.

Genesereth, Michael R. and Nilsson, Nils J. 1987. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, Los Altos, CA.

Genesereth, Michael R. 1988. Introspective fidelity. In *Meta-Level Architectures and Reflection*. Elsevier Science Publishers B.V. (North Holland). 75–86.

Giunchiglia, Fausto and Walsh, Toby 1991. A theory of abstraction. Submitted to Journal of Artificial Intelligence.

Hayes, Patrick J. 1973. Computation and deduction. In *Proceedings of the 1973 Mathematical Foundations of Computer Science Symposium, Czechoslovakian Acadamy of Sciences.*

Kifer, M. 1988. On safety, domain independence, and capturability of database queries. In *Proceedings of the International Conference on Data and Knowledge Bases, Jerusalem.*

Knoblock, Craig; Tenenberg, Josh D.; and Yang, Qiang 1991. Characterizing abstraction hierarchies for planning. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, Cambridge MA. MIT Press. 692–697.

Knoblock, Craig A. 1990. Learning abstraction hierarchies for problem solving. In *Proceedings of the*

Eighth National Conference on Artificial Intelligence, Los Altos, CA. Morgan Kaufmann.

Levy, Alon Y. and Sagiv, Yehoshua 1992. Constraints and redundancy in datalog. In *To appear in the Proceedings of the Eleventh ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, San Diego, CA*.

Levy, Alon Y. 1992. Creating abstractions using relevance claims. In preparation.

Plaisted, D. 1981. Theorem proving with abstraction. In *Artificial Intelligence*. Vol. 16, pp. 47–108.

Sagiv, Yehoshua 1988. Optimizing datalog programs. In Minker, Jack, editor 1988, *Foundations of Deductive Databases and Logic Programming*. Morgan Kaufmann, Los Altos, CA. 659–698.

Shmueli, Oded 1987. Decidability and expressiveness aspects of logic queries. In *Proceedings of the 6th ACM Symposium on Principles of Database Systems*. 237–249.

Smith, David E. and Genesereth, Michael R. 1985. Ordering conjunctive queries. In *Artificial Intelligence*. 26(2) pp. 171–215.

Subramanian, D. and Genesereth, M.R. 1987. The relevance of irrelevance. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Los Altos, CA. Morgan Kaufmann.

Subramanian, Devika 1989. A theory of justified reformulations. In *Ph.D thesis, Dept. of Computer Science, Stanford University*. Stanford, CA.

Tenenberg, Josh D. 1987. Preserving consistency across abstraction mappings. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Los Altos, CA. Morgan Kaufmann. 1011–1014.

Tenenberg, Josh D. 1990. Abstracting first order theories. In Benjamin, Paul, editor 1990, *Change of Representation and Inductive Bias*. Kluwer, Boston, Mass.

Ullman, Jeffery D. 1989. *Principles of Database and Knolwedge-base Systems, Volume I, II*. Computer Science Press, Rockville MD.