

University of Umeå
Department of Computing Science
SE-901 87 Umeå, Sweden

UMINF-02.05
ISSN-0348-0542
May 2002

Z-log: A System-Z (and \hat{Z}) 'System'

by

Michael Minock and Hansi Kraus
mjm@cs.umu.se c97hks@cs.umu.se

ABSTRACT

We present Z-log – a practical system that employs the system-Z[15] semantics. Z-log incurs polynomial cost for compilation and entailment in the horn and q-horn[4] cases. Z-log’s complexity is intractable in the unrestricted case – but intractable in the number of defaults that cause the violation of the q-horn property. In addition we propose an alternative NP-hard compilation strategy which takes $O(3^m)$ space but enables $O(m)$ time entailment for m proposition rule-bases.

We extend the system-Z semantics to what we are calling \hat{Z} . Intuitively \hat{Z} modifies the Z ranking to “increase incredulity” while maintaining the necessary ordering constraints induced by the system-Z partitioning algorithm. \hat{Z} sanctions inheritance of stable properties to exceptional subclasses and preserves ambiguities. Unlike other extensions to system-Z, \hat{Z} maintains q-horn tractability.

We are applying Z-log in cognitive disease diagnosis. This points us toward additional research and development tasks in explanation and knowledge acquisition.

1 Introduction

Efforts have revolved around extending KLM[12] equivalent semantics to more practical *adventurous* semantics. The semantics system-Z [15], conditional entailment[7], maximum entropy entailment[10][5] and lexicographical entailment[13][2] are distinct adventurous semantics.

Although these semantics exhibit many desirable features of common sense, in general they have rather high computational cost. For example, in the general case, entailment in system-Z[8] requires $O(\log(n))$ satisfiability tests, once $O(n^2)$ satisfiability tests are issued to compile a knowledge base of n -defaults. And system-Z is the computationally least expensive semantics of the aforementioned and the only semantics that is tractable in the horn and q-horn cases[6].

However system-Z does not sanction property inheritance across exceptional sub-classes. Although lexicographical and maximum entropy semantics solve this problem, they fail to adequately preserve ambiguities and they are intractable in the q-horn as well as the more restrictive standard horn case. Conditional entailment, though seemingly the most ‘common sense’, presents an enormous computation burden. Because of this, we present an alteration of system-Z which accounts for inheritance of stable properties over exceptional subclasses while of negligible greater computational cost than system-Z.

Although there have been prototype implementations of system-Z[9][5], and an exact counterpart to system-Z using possibility measures[3], Z-log represents the most advanced direct system-Z implementation to date. For example DRS[5] is a semantically based system, calculating an actual world ranking relation over all possible worlds. As such it is only applicable to systems of approximately 16 propositions and 20 defaults. To be fair the intention of DRS was to explore the virtues of Maximum Entropy with variable strength rules, not to build a practical system-Z ‘system’.

1.1 Organization of this paper

In section 2 we shall review system-Z and shall describe the slight alteration to the semantics to achieve what we believe to be more common sense behavior. In section 3 we shall discuss Z-log - our Java implementation of system-Z (and \hat{Z}). In particular we describe our optimization techniques and give some initial performance results. In section 4 we shall touch, in brief, on a real world application in cognitive disease diagnosis. This application points toward problems in explanation generation and knowledge acquisition that must be addressed before fielding Z-log in medical decision support applications.

2 System-Z

System-Z appeals to probability theory to set the meaning of default rules[1]. That is to say the default *cheese* \rightarrow *tasty* means that $P(\textit{tasty} \mid \textit{cheese}) \geq 1 - \epsilon$ where ϵ is an very small value approaching 0.

A world ω is a truth assignment to a set of l propositional variables. A world ranking κ is a function from the 2^l set of worlds Ω to the natural numbers Z : $k(\Omega) \rightarrow Z$. This κ function indicates our surprise with observing a particular world: $P(\omega) \leq \epsilon^{\kappa(\omega)}$. Note that in the case that we are not at all surprised by ω we are simply told that $P(\omega) \leq \epsilon^{\kappa(\omega)} = \epsilon^0 = 1$. This world ranking function is extended to the domain of arbitrary propositional formulas φ .

$$\kappa(\varphi) = \begin{cases} \min_{\omega \models \varphi} \kappa(\omega) & \text{if } \varphi \text{ satisfiable} \\ \infty & \text{otherwise} \end{cases} \quad (1)$$

Given a set of n defaults Δ where the i -th default¹ $\varphi_i \rightarrow \phi_i \in \Delta$ means that when a world satisfies φ_i it will typically also satisfy ϕ_i . This is captured in the following admissibility requirement on κ :

$$\varphi \rightarrow \phi \Leftrightarrow \kappa(\varphi \wedge \phi) < \kappa(\varphi \wedge \neg\phi) \quad (2)$$

There are many world rankings κ which satisfy the above admissibility requirements for a given set of defaults Δ . Let us consider however the most compressed admissible ranking. In this ranking $\kappa(\varphi)$ attains the minimal value that it can under the constraints in Δ . In some sense this is an invitation to accommodate the violation of defaults. It says that whenever we might find the formula α in the violation of $\varphi_i \rightarrow \phi_i$ let us accept this with surprise, but minimal surprise.

The reason we can't simply conclude that all formulas that falsify any default are of surprise 1², is because of the interaction of defaults. For example consider the default $cheese \rightarrow tasty$ and $cheese \wedge dirty \rightarrow \neg tasty$. Note that we must have $\kappa(cheese \wedge dirty \wedge tasty) = 2$. This provides the intuition behind why the most compressed ranked partition of defaults is constructed in system-Z:

Algorithm 1 *Minimal Ranked Partition*

Input: A knowledge base $\Delta = \{\varphi_i \rightarrow \phi_i : 1 \leq i \leq n\}$

Output: An ordered partition of $\Delta = \Delta_1, \Delta_2, \dots, \Delta_m$ ³

begin

$i = 1$;

while $\Delta \neq \emptyset$ **do**

Find the set of defaults $\Delta_i = \{\varphi_j \rightarrow \phi_j\}$ from Δ ,
such that $\{\varphi_j \wedge \phi_j\} \cup \Delta$ is satisfiable.

if $\Delta_i = \emptyset$ **then** abort Δ is inconsistent.

else remove Δ_i from Δ and set $i = i + 1$;

fi

od

Return $\Delta = \Delta_1, \Delta_2, \dots, \Delta_m$;

end

¹The extension of system Z to handle strong rules in the company of defaults is straight forward and will be left out of our review.

²This indeed would be the maximally compressed ranking that still recognized default violations as surprising.

³Note that our numbering of the partition levels starts at 1 instead of 0 as in [8].

Under such a ranked partition, $\kappa_z(\alpha)$ is the highest partition in which α falsifies a default while in the company of the defaults in that partition and partitions above it. This gives us the entailment semantics for system Z: $\varphi \vdash_z \phi$ iff $\kappa_z(\varphi \wedge \phi) < \kappa_z(\varphi \wedge \neg\phi)$.

2.1 \hat{Z}

A default will be placed in the first partition Δ_1 , when it has the property that if it is verified⁴ it does not falsify any other default in the knowledge-base. Note also that a default that overrides another, must be placed in a higher partition than the first. Hence a world which falsifies an override is seen as more surprising than a world that falsifies a non-overridden default. But this runs counter to common sense. If anything we should be more surprised by a violation of a non-overridden default than by the violation of an override. The mere presence of an override signals that the material over which the default applies is subject to more non-monotonic behavior.

These considerations lead us to a modification of system-Z that treats the violations of non-overridden defaults with greater incredulity, while preserving the necessary ordering properties embedded in the most compressed admissible ranking built by Z.⁵

2.1.1 Pinned Defaults

If a default is not “pinned” to a partition, that default should be promoted to a higher partition, thus increasing our incredulity in the case that a world were to falsify such a default.

Definition 1 A default $\varphi_i \rightarrow \phi_i \in \Delta_k$ is pinned at level k if either

- (a) $\exists \varphi_j \rightarrow \phi_j \in \Delta_{k+1}$ where $\varphi_j \cup \phi_j \cup \Delta_{k+1} \cup \Delta_k - \{\varphi_i \rightarrow \phi_i\}$ is a satisfiable set of formulas
- (b) $\exists \varphi_{j'} \rightarrow \phi_{j'} \in \Delta_{k+1}$ where $\varphi_{j'} \cup \phi_{j'} \cup \Delta_{k+1} \cup \varphi_i \rightarrow \phi_i$ is a non-satisfiable set of formulas.

The intuition behind pinning is that either the default’s promotion would cause a higher ranked default to no longer be required to be at its minimal Z determined rank - case (a). Or the default itself should not be promoted because that would lead to a case of inconsistency within the higher rank - case (b). Note that condition (a) guarantees that there is at least one pinned default per partition level and that the set of rules that will promote will be unique.

2.1.2 Default Promotion Algorithm

The following algorithm that accepts a Z-partition and builds the \hat{Z} partition where unpinned defaults are promoted to higher partition levels:

⁴Both its antecedent and conclusion are true.

⁵Variable strength defaults complicate \hat{Z} . Thus far we have not considered the precise effect that such defaults would have on the \hat{Z} .

Algorithm 2 *Minimal Pinned Partition*

Input: A Z ordered partition of $\Delta = \Delta_1, \Delta_2, \dots, \Delta_m$

Output: An ordered partition of $\hat{\Delta} = \hat{\Delta}_1, \hat{\Delta}_2, \dots, \hat{\Delta}_m$ where $\hat{\Delta}_i, i < m$ consists of only defaults pinned by the original partition Δ .

begin

$i = 1;$

while $i < m$ **do**

Promote the non-pinned defaults in Δ_i to Δ_{i+1} .

set $i = i + 1;$

od

end

Theorem 1 *The \hat{Z} partition is unique and consists of m -levels.*

The proof follows immediately from the algorithm and properties of the input Z-partition.◊

Note that the procedure to compute whether $\varphi \rightarrow \phi$ follows from Δ is the same as that used in system-Z but with the new, pinned partition. Hence the cost for entailment after compilation is $\log(n)$ calls to a SAT solver. While the \hat{Z} partition may be built in $O(n^3)$ calls to a SAT solver.

2.2 Improvements, Limitations, and Extensions

\hat{Z} provides an entailment semantics apparently equivalent to that in the *free defaults* proposal of [3]. Consider the example in [6]: $penguin \Rightarrow bird, bird \rightarrow fly, penguin \rightarrow \neg fly, bird \rightarrow wings, penguin \rightarrow arctic, fly \rightarrow mobile$, system-Z handles transitivity ($bird \rightarrow mobile$), irrelevant information ($bird \wedge red \rightarrow fly$), and overrides ($penguin \rightarrow \neg fly$). However it does not sanction inheritance over exceptional subclasses ($penguin \rightarrow wings$). If we now calculate the minimal pinned ranking we obtain $\hat{\Delta}_1 = \{bird \rightarrow fly\}$ and $\hat{\Delta}_2 = \{penguin \rightarrow \neg fly, penguin \rightarrow arctic, bird \rightarrow wings, fly \rightarrow mobile\}$. Thus $\kappa(penguin \wedge wings) = 1 < \kappa(penguin \wedge \neg wings) = 2$ and thus ($penguin \rightarrow wings$).

Maximum entropy[10], lex-entailment[13][2] and conditional entailment[7], likewise entail ($penguin \rightarrow wings$). However if we add $propeller \rightarrow fly$, and $light \rightarrow fly$. If we consider whether a light penguin with a propeller should be deduced to fly or not, common sense tells us that there is not enough information to conclude the matter either way⁶. However under Z and lex-entailment we conclude that such a penguin will not fly. Under maximum entropy we conclude that such a penguin will fly. Only under conditional entailment, ϵ semantics, and \hat{Z} is the proper behavior observed. Since ϵ semantics is weak and conditional entailment expensive, we opt for \hat{Z} .

Unfortunately \hat{Z} fails to sanction the inheritance of non-stable properties to exceptional subclasses. Consider $cheddar \Rightarrow cheese, swiss \Rightarrow cheese, cheese \rightarrow tasty, cheese \rightarrow yellow, cheddar \rightarrow \neg yellow$, and $swiss \rightarrow \neg tasty$. Unfortunately there is no way to build ranked

⁶In essence we have yet another example of the Nixon diamond.

partitions such that we keep swiss cheese yellow, but not tasty, and we keep cheddar tasty, but not yellow. Such considerations seem to doom approaches based on linear rankings of partitions.

One possible remedy that we are entertaining is to build a tree of alternative rankings of depth m . And to copy defaults that are not pinned (by criteria b) up the branches of the tree toward the leaves. Though in the worst case this yields a tree with a combinatorial number of paths from leaf nodes to the root, in practice the number of seems to be limited. These paths may then be treated independently as a set of ranked partitions $\Delta^1, \Delta^2, \dots, \Delta^p$. Furthermore we enrich the notion of $k_z(\alpha)$ to $k'_z(\alpha)$, where a rank value may be further reduced if partitions under the standard $k_z(\alpha)$ rank are unsatisfiable in the company of α . We then calculate the final value of $k(\alpha)$ to be the maximum value that $k'_z(\alpha)$ obtains over all the rankings $\Delta^1, \Delta^2, \dots, \Delta^p$. For the cheese example above we have: $\Delta_1^1 = \Delta_2^1 = \{\textit{cheese} \rightarrow \textit{tasty}, \textit{cheese} \rightarrow \textit{yellow}\}$, $\Delta_2^1 = \{\textit{cheddar} \rightarrow \neg\textit{yellow}, \textit{cheese} \rightarrow \textit{tasty}\}$, and $\Delta_2^2 = \{\textit{swiss} \rightarrow \neg\textit{tasty}, \textit{cheese} \rightarrow \textit{yellow}\}$. $k(\textit{cheddar} \wedge \textit{tasty}) = 0 < k(\textit{cheddar} \wedge \neg\textit{tasty}) = 1$ and $k(\textit{cheddar} \wedge \neg\textit{yellow}) = 1 < k(\textit{cheddar} \wedge \textit{yellow}) = 2$.

This proposed remedy holds some promise. The semantic notions are reminiscent of conditional entailment[7] but entailment queries leverage off of the linear ranked partition engine we have already built in Z-log. However the implementation of these notions is still being undertaken. In the meantime we take the limited inheritance over exceptional subclasses that \hat{Z} affords us. And we take some solace in the fact that $\kappa_{\hat{z}}$, unlike κ_z , treats the falsification of non-overridden defaults with greater surprise than the falsification of overridden defaults.

3 Implementation: Z-log

We are calling our implementation of the system-Z (as well as \hat{Z}) Z-log. After an initial LISP prototype, we build Z-log in 15 Java classes written in approximately 4000 lines of code. The system will soon be available for down-load from <http://www.cs.umu.se/~mjm>. The system is run by providing a definition file (.def) in which strong and default propositional rules are specified and a query file (.qry) where facts are asserted (or retracted) and a stream of entailment and κ queries are processed.

The following is the definition file `catAndMouse.def`.⁷

```

/* We may (optionally) declare propositions before we use them with '+'. */
+cheese cheddar tasty dirty . // Features of the cheese.
+starving seekCheese. // the mouse.
+catAtHome catInKitchen catInDen, catInLivingRoom catInBedroom.// the cat
+night. // and the world.

/* The defaults: prolog style, order unimportant */
cheese := cheddar. // A hard rule.
tasty :- cheese. // A default.
!tasty :- cheese & dirty.
tasty :- cheese & dirty & starving.

!catInKitchen := catInDen.
!catInLivingRoom := catInDen.

```

⁷Note the PROLOG like syntax. The default $\neg a \wedge b \rightarrow c$ is written `c :- !a & b`. The hard rule $a \Rightarrow b \vee c$ is written `b | c := a`. Comments are as in C.

```
!catInLivingRoom := catInKitchen.
!catInBedroom := true. // false is also a built-in proposition.

catInKitchen | catInLivingRoom | catInDen :- catAtHome.
catInDen :- night.
seekCheese :- !catInKitchen & cheese.
seekCheese :- starving & cheese.
!seekCheese :- catInKitchen.
```

The following is a query file that corresponds to a fact context and a stream of queries over the definition file `sylvester.qry`.

```
chedder.
!catInLivingRoom.
?tasty. // Should be TRUE.
?catInKitchen. // Should be UNDECIDED.
?!catInBedRoom. // Should be TRUE.
?seekCheese. // Should be UNDECIDED.
night.
?catInKitchen. // Should be FALSE.
?seekCheese. // Should be TRUE.
dirty.
?tasty. // Should be FALSE.
starving.
?tasty. // Should be TRUE.

/* Kappa value queries */
#cheese & !tasty. // Should be 1.
#night & !catInDen. // Should be greater than 1.
```

At the command line one types:

```
java Zlog -Z^ catAndMouse.def sylvester.qry
```

This sends to standard output the results from compiling `catAndMouse.def` and then processing all the queries in `sylvester.qry` under \hat{Z} semantics. Note that a `catAndMouse.par` file is written to disk so Z-log does not need to recompute the partition over multiple program invocations.

Under Z and \hat{Z} all the queries are answered correctly. However \hat{Z} registers more surprise in a world where the cat is not in the den at night, than a world where cheese is not tasty.

3.1 Optimization Techniques

Z-entailment has been shown to be P-complete for several syntactic restrictions of knowledge-bases. Notably the horn case, and the more recent q-horn case[6]. Certainly if knowledge bases arrive in these forms we would like our algorithms to indeed solve these problems in polynomial time. However we are interested being able to use Z-log to solve queries over any syntactically well formed definition file. And this means that Z-log must be able to handle the combinatorial cases.

3.1.1 The Core Satisfiability Algorithm

At the core of the compilation and inference algorithms, we are concerned about whether a set of clauses are satisfiable given a conjunct of literals. Because of our intention to compile κ *Index Structures* (see below), we decided against using the classical backtracking algorithm in favor of an approach that keeps a DNF expressing satisfiable worlds. Importantly when using this strategy we are able to still compute satisfiability in P over the q-horn case.

Let us illustrate the core operation of our algorithm. Given a conjunct of literals $l_1 \wedge \dots \wedge l_m$ we progress through our set of rules expressed as clauses. For each rule we 'subtract' the worlds that falsify the rule. Thus if we pick the rule $l'_1 \wedge \dots \wedge l'_{m'} \rightarrow l''_1 \vee \dots \vee l''_{m''}$ then when we 'subtract' the worlds that falsify this clause from our input conjunct, we generate the DNF: $(l_1 \wedge \dots \wedge l_m \wedge \neg l'_1) \vee \dots \vee (l_1 \wedge \dots \wedge l_m \wedge \neg l'_{m'}) \vee (l_1 \wedge \dots \wedge l_m \wedge l''_1) \vee \dots \vee (l_1 \wedge \dots \wedge l_m \wedge l''_{m''})$

Each conjunct in the resulting DNF now describes a set of worlds permitted after we consider the first rule in the company of our input conjunct. When we consider the next rule, we must subtract the worlds that falsify that rule from each of the conjuncts in the DNF expression.

If this ultimately yields an empty expression, then the rules are unsatisfiable with respect to our input conjunct. If, however we yield a DNF expression at the end of this process then the conjunct is satisfiable with respect to our rules. Moreover if a literal l is within every conjunct of that DNF, then that literal is a consequence of the original conjunct in the company of our rules.

3.1.2 The Pure Horn Case

Clearly if no heuristics govern the above procedure, namely which rule to pick next, then we are in for a combinatorial nightmare. However consider the pure horn case where each rule may be represented as a clause with at most 1 positive literal. In this case we apply only rules that will result in a single conjunct DNF. This property simply means that the rule of m literals has $m - 1$ of its negative literals in their positive form within the single conjunct DNF. If rules remain to be processed, yet none have this property, we are then guaranteed that the rules are satisfiable with respect to our input conjunct. If during our processing of rules, we generate an empty DNF, then we know that the set of rules is unsatisfiable in the company of the input conjunct. Observing this algorithm in action illustrates that it just mimics the standard bottom up evaluation strategy for horn clauses applied over a given set of facts.

3.1.3 The q-horn Case

Note that through renaming one may often take a set of non-horn rules and make them horn. This is achieved by identifying literals such as p which only appear in positive form in the clauses. By renaming p to $\neg \bar{p}$ we are able to reduce by one the number of positive literals in the clauses where p was present. If we can repeat this process, then we may in fact reduce our rules to a set of horn clauses. Such rules are said to be in *disguised horn form*.

However the q-horn case is more broad⁴. In the q-horn case: 1.) we may build a partial assignment S to literals such that S may include either p or $\neg p$ but not both; 2.) A clause may have at most two literals not within S ; 3.) When a clause has exactly two literals t_1 and t_2 not in S then neither $\neg t_1$ nor $\neg t_2$ are in S ⁸.

⁸We shall say that the set of propositions with such a property are the set T

As a practical matter S serves as our guide to do a renaming where all the positive literals p in S are consistently renamed to \bar{p} (therefore $\neg\bar{p} \in S$). Therefore, after renaming, our new S consists entirely of negative literals, and hence we may rewrite our set of clauses as a set of clauses each with at most 2 positive literals. Furthermore in the case of exactly two positive literals - both will be over propositions within T . Let us assume that such a renaming has occurred throughout the rest of our exposition.

This leads us to the following observation about the rules represented in our clauses. There are three possible types for the horn case: (1) $s_i \wedge \dots \wedge s_j \rightarrow s_k$, (2) $s_i \wedge \dots \wedge s_j \rightarrow \neg s_k$, and (3) $s_i \wedge \dots \wedge s_j \rightarrow t_k$. And there is one type (4) for the form of our non-horn clauses: $s_i \wedge \dots \wedge s_j \rightarrow t_{k_1} \vee t_{k_2}$ ⁹.

Now we formulate a strategy where if we can not find a suitable rule to apply, then we may quickly decide whether the original conjunct is satisfiable with respect to our set of rules.

Definition 2 *Given a conjunct ϕ , a clause of length m is **suitable** if at least $m - 1$ of its literals appear in their reversed form in ϕ .*

Of course the *application* of a suitable rule to a conjunct, results in either cancellation or a single resulting conjunct.

Lemma 1 *Given a set of clauses in q -horn form, renamed such that S consists of entirely negative literals, if no suitable clauses may be applied to ϕ then $\phi \wedge s$ is satisfiable where $s \in S$.*

Proof: Because no suitable clauses may apply to ϕ then all m length clauses that apply to $\phi \wedge s$ must contain the literal $\neg s$, thus must be of type 2, and must have $m - 2$ of it's literals in reversed form within ϕ . Any number of applications would result in literals such as $s' \in S$ being added to the conjunct ϕ . Such applications lead to cancellation, and thus unsatisfiability, only if s'' is derived where $\neg s''$ is within ϕ . Let us assume that this were the case. This would mean we could apply the clause $s_1 \vee \dots \vee s_{m-2} \vee s'' \vee \neg s'$. This would mean that $\neg s_1 \dots \neg s_{m-2}$ must be in ϕ . And this would only cancel if $\neg s''$ were also in ϕ . But if this were the case then we would have been able to apply the clause to ϕ to generate $\phi \wedge \neg s'$ in the first place, thus violating our assumption that no suitable clause could apply to ϕ . \diamond

Lemma 2 *Given a set of clauses in q -horn form, renamed such that S consists of entirely negative literals, if no suitable clauses may be applied to ϕ then we may decide in polynomial time whether $\phi \wedge t$ (or $\phi \wedge \neg t$) is satisfiable where $t \in T$.*

Proof: We may conclude that the initial rules that may be suitable to $\phi \wedge t$ are of types 3 or 4. An application of a type 3 rule would result in a literal s being introduced into the conjunct. The same argument in Lemma 1 may be applied to argue that this will not lead to cancellation. In the case of an application of a rule of type 4 this would lead to another t' being added to $\phi \wedge t$. This would only lead

⁹ t_k is a literal over a proposition in T and $\neg s_i \in S$, $\neg s_j \in S$ and $\neg s_k \in S$.

to cancellation if the literal $\neg t'$ could be added after subsequent applications of rules of type 4. Since it is impossible that additional $\neg s$ literals will be added to ϕ , we need only to consider whether applications of type 4 rules may lead to a cancellation. This means we need to only consider a finite set of 2 literal disjuncts from T are satisfiable given t . This is of course a (very) simple 2SAT problem. If it is satisfiable then $\phi \wedge t$ may not cancel. Otherwise it will. \diamond

The following theorem lets our algorithm decide satisfiability for q-horn clauses in polynomial time.

Theorem 2 *Given a conjunct ϕ , and a set of clauses in q-horn form, renamed such that S consists of entirely negative literals, if no suitable clause may apply to ϕ and no-type 4 rule can split the conjunct into two unsatisfiable pairs, then ϕ is satisfiable.*

Proof: Assume that for ϕ does not have any suitable rules which apply to it. If we apply a type 1 rule (such as $s \rightarrow s'$) we will generate the DNF: $(\phi \wedge \neg s) \vee (\phi \wedge s')$. By lemma 1 the first conjunct is satisfiable. If we apply a of type 2 rule (such as $s \rightarrow \neg s'$), then we create the DNF: $\phi \neg s \vee \phi \neg s'$. By lemma 1 both conjuncts are satisfiable. If we apply a type 3 rule (such as $s \rightarrow t'$), we create $(\phi \wedge \neg s) \vee (\phi \wedge t)$. The first conjunct may not cancel. Finally if we apply a rule of type 4 we generate either the DNF: $(\phi \wedge \neg s') \vee (\phi \wedge t)$ or we generate the DNF $(\phi \wedge t) \vee (\phi \wedge t')$. The first case is satisfiable based on lemma 1. From lemma 2, we may determine in polynomial time whether the DNF $(\phi \wedge t) \vee (\phi \wedge t')$ will or will not cancel out. We need to do this over each possible type 4 rule that have their antecedent literals in reversed form in ϕ , but this we do in deterministic polynomial time. \diamond

3.1.4 Online Approach to the Combinatorial Case

In the combinatorial case it is first important to build a subset of the KB that is in q-horn form. Given this the remaining n' defaults which lie outside this set are applied to the supplied conjunct. This will yield a DNF with on the order of $O(2^{n'})$ conjuncts. These conjuncts in turn will then each undergo polynomial time treatment with the respect to that portion of the KB that was identified as being in q-horn form. **Thus the complexity is intractable on the order of the size of the non q-horn portion of the KB.**

3.1.5 Compilation to off-line, NP-Space Structure

Although we have not yet implemented the notion, we recognize that our strategy of generating the complete DNF that describes satisfiable worlds can be utilized to take knowledge compilation to the extreme. Because a formula in DNF may be tested for satisfiability in $O(n)$ time, we convert the CNF defining the k -th most surprising worlds into a DNF expression¹⁰. This expression in turn is combined with the DNF describing the $k - 1$ -th most surprising worlds. This is repeated resulting in a exponentially sized index-like structure that may be queried in $O(m)$ time where m is the number of attributes. We are terming this the κ *index structure*.

¹⁰Through setting the input conjunct to *true* and then applying all rules in $k - th$ partition and above.

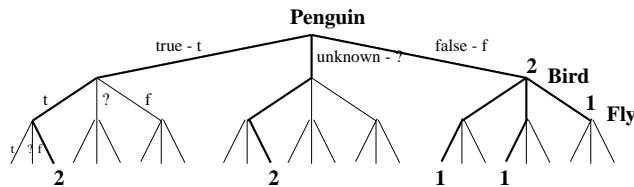


Figure 1: κ index structure

To clarify this consider the default-base $\Delta = \{penguin \Rightarrow bird, bird \rightarrow fly, penguin \rightarrow \neg fly\}$. This is partitioned into $\Delta_1 = \{bird \rightarrow fly\}$ and $\Delta_2 = \{penguin \rightarrow \neg fly\}$. Thus the DNF that describes those worlds of surprise 2 or less is $(\neg penguin) \vee (bird \wedge \neg fly)$ and the DNF that describes worlds that are of surprise 1 or less is $(\neg penguin \wedge \neg bird) \vee (\neg penguin \wedge fly)$. Figure 1 shows these DNFs embedded in a tri-nary tree structure.

The k value of that conjunct may be computed by accessing worst case m nodes in this tri-nary tree. The cost associated with building this index structure is payed during partition construction. The cost continues in the form on having to store an exponential space structure¹¹. But the speed of online deduction becomes $O(m)$ and may be calculated using a very light weight C program.

3.2 Performance

We show in the figure below our runtime costs for Z partition construction on the Y-axis and our problem space size in number of defaults on the X axis. The actual problem we are running is a simplified, non-deterministic version of Wumpus world. Non-deterministic in that forward actions may or may not be performed, and simplified in that the world is an array of black and white tiles. Although this is, no doubt, a toy problem, such a spatial-temporal reasoning problem generates many hundreds of propositional rules for even small array sizes. In the future we intend to chart the performance of Z-log over a wider class of benchmarks.

4 Medical Decision Support - Cognitive Disorder Diagnosis

We believe that Z-log may be of utility within a medical decision support tool. *Medical decision support systems* give advice to medical personal. Such advice is relative to the patients complex and often incomplete medical record. Published clinical guidelines serve as a knowledge source from which to author initial defaults and hard constraints. The knowledge-base is then refined by medical personnel to capture unwritten knowledge of the domain.

¹¹Only the darkened branches in the above example would need to be stored.

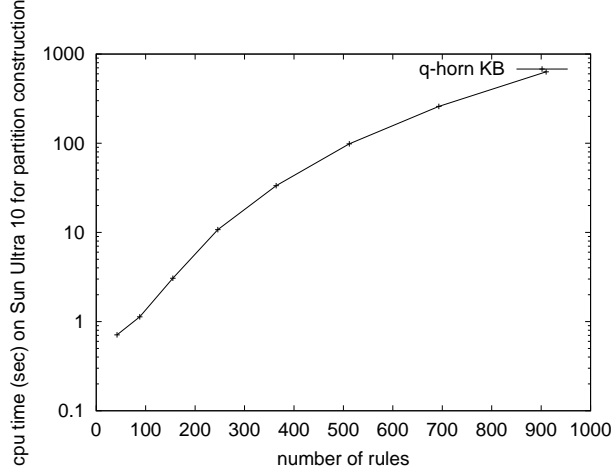


Figure 2: Initial performance result

We are currently building a Z-log knowledge-base devoted to the problem of computing cognitive disorder diagnosis. Our knowledge was obtained from clinical guidelines and discussions with physicians[14]. Currently we have a system with approximately 100 hard rules and 30 defaults. The advice that the system generates seems sound. However before we progress on to a field experiment, with feedback and criticism from physicians, we feel that it is necessary to address some problems in knowledge acquisition and explanation.

4.1 Explanation

Explanation is very important in knowledge-based system. And this is especially true for medical decision support. Currently we have some experimental code that computes explanations¹².

Definition 3 *The triple (φ', Δ', L') is an **explanation** of $\varphi \vdash_z \phi$ under L iff $\Delta' \subseteq \Delta$, $\varphi \models \varphi'$, $L' \subseteq L$ and $\varphi' \vdash_z \phi$ in the knowledge base Δ' under L' .*

The above simply states the Deductive-Nomological account of explanation[11] for system-Z.

Definition 4 *An explanation $e = (\varphi', \Delta', L')$ of $\varphi \vdash_z \phi$ is **minimal** if $\neg \exists e' = (\varphi'', \Delta'', L'')$ such that e' is an explanation of $\varphi \vdash_z \phi$ and $\varphi' \models \varphi'' \wedge \Delta'' \subseteq \Delta' \wedge L'' \subseteq L' \wedge e \neq e'$*

Minimal explanations include only non-superfluous facts, defaults, and rules.

Definition 5 *An explanation $e = (\varphi', \Delta', L')$ of $\varphi \vdash_z \phi$ is in the **context** of φ, Δ , and L if $\forall \varphi'' \forall \Delta'' \forall L'' (\varphi \models \varphi'' \models \varphi' \wedge \Delta' \subseteq \Delta'' \subseteq \Delta \wedge L' \subseteq L'' \subseteq L \Rightarrow (\varphi'', \Delta'', L''))$ is an explanation of $\varphi \vdash_z \phi$.*

¹²The explanations that are computed by this system make no distinction between causal and evidence defaults. Hence the explanations are only epistemic and should be interpreted as forming a justification of why the Z-log believes certain propositions given that it believes others.

Finally, because of the default nature of our knowledge-bases, we must isolate our attention to explanations that are in the same context as the original deduction.

Thus far the algorithms that we have considered have all been doubly exponential. However we see some potential in compiling explanations into the κ index structure.

4.2 Dialog-based Knowledge Acquisition

As stated in the introduction, Z-log enables defaults to be provided that do not explicitly require exceptions to be mentioned. This makes defaults much more compact and enhances explanation generation. It also assists in knowledge acquisition.

The protocol envisioned here is one in which the expert observes system inferences and then corrects system mistakes, thus refining the knowledge-base. There exist six-basic correction strategies based on the combination of TRUE, FALSE, and UNDECIDED answer to what in fact should be TRUE, FALSE, or UNDECIDED. The 4 cases where a result should be definite (TRUE or FALSE) call for additional default rule(s) to be inserted into the knowledge-base. The two cases where the result should be UNDECIDED calls for counter factual feedback to reintroduce ambiguity. Finally the system should be able to generate questions to the expert while it tries to simplify its knowledge structures or resolve inconsistencies in teaching. Question may also be generated to patch the rule system to fully represent inheritance over exceptional subclasses.

5 Conclusion

We have described the first implementation of Z-log: a q-horn tractable system that encodes system-Z[15] and \hat{Z} . We have provided initial performance results that show the system is able to scale to q-Horn form knowledge-bases numbering in the thousands of rules. In the future we intend to more precisely characterize the knowledge bases and types of problems over which Z-log may be practically applied. We also intend to further seek out approaches that give Z-log the capability to fully sanction property inheritance over exception subclasses.

Ultimately Z-log is meant to be used by individuals who have little or no experience in knowledge engineering. However to achieve real use in medical decision support we understand that very difficult and computationally challenging problems must be solved in explanation generation and knowledge acquisition over Z-log.

References

- [1] Ernest Wilcox Adams. *The Logic of Conditionals*. D. Reidel, Dordrecht, 1975.
- [2] S. Benferhat, C. Cayrol, D. Dubois, J. Lang, and H. Prade. Inconsistency management and prioritized syntax-based entailment, 1993.

- [3] Salem Benferhat, Didier Dubois, and Henri Prade. Representing default rules in possibilistic logic. *KR 1992: 673-684*, 1992.
- [4] E. Boros, Y. Crama, and P. Hammer. Polynomial-time inference of all valid implications for horn and related formulae. *Annals of Mathematics and Artificial Intelligence*, 1:21–32, 1990.
- [5] R. Bourne and S. Parsons. Maximum entropy and variable strength defaults, 1999.
- [6] Thomas Eiter and Thomas Lukasiewicz. Complexity results for default reasoning from conditional knowledge bases. In *KR2000: Principles of Knowledge Representation and Reasoning*, pages 62–73, San Francisco, 2000. Morgan Kaufmann.
- [7] H. Gefner. Default reasoning: causal and conditional theories, 1992.
- [8] Goldszmidt and Judea Pearl. Qualitative probabilities for default reasoning, belief revision, and causal modeling. *Artificial Intelligence*, 84(1-2):57–112, 1996.
- [9] M. Goldszmidt. *Qualitative Probabilities: A Normative Framework for Common-sense Reasoning*. PhD thesis, University of California, California, 1992.
- [10] Moisés Goldszmidt, Paul Morris, and Judea Pearl. A maximum entropy approach to nonmonotonic reasoning. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, Menlo Park, CA, 1990. AAAI Press.
- [11] C. Hempel. Explanation in science and history. In R. Colodny, editor, *Frontiers in Science and Philosophy*, pages 7–34. University of Pittsburg Press, 1962.
- [12] S. Kraus, D. Lehmann, and M. Magidor. Preferential models and cumulative logics. *Artificial Intelligence*, 44:167–207., 1990.
- [13] Daniel J. Lehmann. Another perspective on default reasoning. *Annals of Mathematics and Artificial Intelligence*, 15(1):61–82, 1995.
- [14] Helena Lindgren. Decision support systems and diagnosing cognitive disorders. Master’s thesis, University of Umeå, Sweden, 2000.
- [15] J. Pearl. A natural ordering of defaults with tractable applications to default reasoning. *Proceedings of the 3rd Conference on Theoretical Aspects of Reasoning about Knowledge*, pages 121–135., 1990.