A QUANTITATIVE ANALYSIS OF REASONING FOR RMS

Laurent Buisson and Jérôme Euzenat

Division Nivologie
CEMAGREF
BP 76
F-38402 SAINT-MARTIN D'HÈRES

IRIMAG/INPG
Laboratoire ARTEMIS/Imag
BP 53 X
F-38041 GRENOBLE Cedex

internet: Jerome.Euzenat@sherpa.imag.fr — uucp: euzenat@imag.fr

ABSTRACT

For reasoning systems, it is sometime useful to cache away the inferred values. Meanwhile, when the system works in a dynamic environment, cache coherence has to be performed, and this can be achieved with the help of a reasoning maintenance system (RMS). The questions to be answered, before implementing such a system for a particular application, are: how much is caching useful ? Does the system need a dynamicity management system ? Is a RMS suited (what will be its overhead) ?

We provide an application driven evaluation framework in order to answer these questions. The evaluation is based on the real work to be processed on the reasoning of the application. First, we express the action of caching and maintaining with two concepts: backward and forward cone effects. Then we quantify the inference time for those systems and find the quantification of the cone effects in the formulas.

1. INTRODUCTION

For reasoning systems such as knowledge bases, it is often necessary to record the result of the inference process even if it is goal driven. Recording the result of a computation is called *caching* in computer science. Caching is necessary when the produced inferences are costly and used several times.

When knowledge in the base does not evolve, caching is safe and very efficient. But in real world applications, the knowledge base is usually dynamic. This is true for systems that interact with the environment (through sensors) or with the user who can set hypotheses and change the knowledge in the base. So, caching requires dynamicity management. Most of the time, it is performed by using a RMS (Reasoning Maintenance System) based on dependency graph manipulation. But is a RMS always interesting ? Should it be more attractive to treat dynamicity problems by ignoring RMS solutions ?

We develop here a quantitative analysis of the reasoning graph in order to answer these questions. Numeric criteria defined on properties of the dependency graph are used. Real world applications give evidence of such properties, especially for spatial knowledge bases and spatial reasoning.

After a short description of reasoning maintenance systems and their advantages in the context of knowledge bases, we will briefly describe an object-based knowledge

base management system called Shirka/TMS which uses a RMS (§2). We will show some numeric results from that system and give expectations about its behavior. More recently, observations have been performed on a real world application ELSA (§3) dedicated to the analysis of snow avalanche path. This application uses the inference mechanisms in order to compute spatial properties of a geographic area such as *connected* sub-areas or *close* ridges which are used very often. The benchmark results obtained with ELSA are very surprising.

We are able to explain them with the help of a new concept: backward and forward cone effects. They are formalized (§4) in order to draw general conclusions about RMS use in reasoning systems. In fact, the advantage of a RMS toward rough caching is a tradeoff between backward and forward cone effect.

2. A SPATIAL REASONING APPLICATION

The motivations for using a RMS in knowledge based systems are first presented. Then, Shirka/TMS will be introduced together with some tests and expectations about its behavior.

2.1. REASONING MAINTENANCE SYSTEM

When using an inference system in backward chaining mode, the result of each inference, would it be an attribute value or the validity status of a proposition, can be cached i.e. recorded in memory. Cached values do not have to be inferred twice or more. In fact, caching is useful when a value is used several times by the system and is as useful as the number of times the value is needed. But, while caching uses additional memory space and time, it has to be used with care.

Moreover, in evolving systems or when the inferences allowed by the system can be nonmonotonic, something which is considered as holding (a value considered as the value of an attribute or a proposition considered as true) can be discarded. In such cases, the cached values must be invalidated, i.e. not cached anymore. This is the job of a RMS.

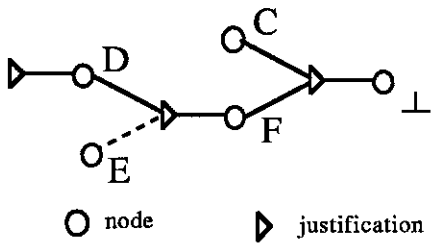

Fig. 1. A dependency graph is here represented with circles as nodes and triangles as justifications where the nodes in the IN-list come through a full line while nodes in the OUT-list come through a doted line. Nodes that have a justification whose IN- and OUT-lists are empty (e.g. D) represent true formulas because they do not need to be inferred.

Reasoning maintenance systems (RMS) are aimed at managing a knowledge base considering different kinds of reasoning. Such a system is connected to a reasoner (or problem solver or inference engine) which communicates every inference made. The RMS has in charge the maintenance of the reasoner's current belief base. RMS developed so far focussed on nonmonotonic reasoning or multiple contexts reasoning. They record each inference in a *justification* that relates *nodes* representing propositional formulas plus a special atom ($\perp$) representing contradiction. A justification ($<\{i_1,\ldots i_n\}\{o_1,\ldots o_m\}>$: c) is made of an IN-list ($\{i_1,\ldots i_n\}$) and an OUT-

list ($\{o_1,...o_m\}$). Such a justification is said to be valid if and only if all the nodes in the IN-list are known to hold while those in the OUT-list are not; a node, in turn, is known to hold if and only if it is the consequent (c) of a valid justification. The recursion of the definition is stopped by nodes without justification and by the axioms that are nodes with a justification containing empty IN- and OUT-lists.

2.2. SHIRKA/TMS AND ITS BEHAVIOR

Shirka is a traditional object based knowledge representation system written in Lisp [1]. Everything, in Shirka, is an object (including inference methods...). Each object belongs to a class which defines its structure — in terms of a list of fields and constraints on the fields values — and its inferential capabilities — in terms of inference methods used in order to determine the values of unfilled fields. Inference methods are among value passing, procedural attachment, pattern matching and default values.

Classes are organized in a direct acyclic graph structured by the *a-kind-of* relationship between classes. This relationship enables inheritance from a class to its specializations. Inheritance is used through class refinement — a class strongly inherits, i.e. possesses, its constraints on fields from its super-class — and inference specialization — a class weakly inherits, i.e. inherits by default, its inference methods.

A RMS has been implemented on Shirka. It is standard except that it records and propagates field values [2]. The underlying assumption of the implementation of a RMS in an object-based knowledge representation is that the base is queried very often (or not often modified). The performances are very attractive because re-infering is avoided (and so, the answers are given very quickly). On another hand, the modifications — that are safely dealt with — and initial inferences are processed more slowly. This assumption was enforced by the observations made with the very simple tests below.

2.3. ELSA: A SPATIAL REASONING APPLICATION

In the context of spatial reasoning, the RMS is very attractive. In other words, spatial reasoning appears as a good application domain. Meanwhile, some effects which have not been presented yet can be observed in that kind of applications: they are "forward and backward cone effects". These observations were performed on a real world application dedicated to the analysis of snow avalanche paths: ELSA.

We first present ELSA and the advantage of using a RMS in the context of spatial reasoning. Then, a set of numeric tests are discussed which demonstrates the advantage of using a RMS in ELSA. At last, those results are summarized in two principles called backward and forward cone effect.

ELSA is a problem solving environment which offers to a snow specialist the different tools available in order to perform an avalanche path analysis and choose the best protection devices. As it has been explained elsewhere [3, 4], ELSA is built on Shirka/TMS. ELSA is a knowledge based system which uses both symbolic simulation based on expert knowledge and numerical simulation based on fluid mechanics conservative laws.

Because of the spatial extension of the phenomena involved in snow avalanches (snow-drift, snow-cover stability, fracture propagation, avalanche flowing...), ELSA needs spatial information on the path. In order to get this information or to use it, ELSA performs an actual spatial reasoning as it has been defined in [3]. As a matter of fact, from poorly relevant spatial knowledge such as contour line, vegetation or ridge maps,

ELSA must infer the definition of special units of terrain called "small panels" by snow specialists and which are relevant for analysis (they are homogeneous from the analysis criteria points of view). Meanwhile this definition in small panels is not relevant enough and ELSA must also infer the properties of these small panels to perform its analysis.

These inferences are taken into account by the knowledge base system. In this paper we emphasize on terrain inference: the inference of spatial relevant properties from poor spatial knowledge. For example, here is the spatial definition of a small-panel called `pp1`. At the beginning of the session, this small panel is defined only by the list of triangles included in it. As it has been written in Shirka, the syntax is frame like.

```
{pp1
        is-a                = small-panel ;
        contains            = tr2 tr93 }
```

In order to make an analysis of the avalanche starting zone, ELSA needs more relevant information and, to that extent, infers a more complete description of the small panel `pp1`. All the fields inferred by ELSA are obtained by the use of inference methods (as presented above), particularly, pattern-matching inference and procedural attachment.

```
{pp1
        is-a                = small-panel ;
        area                = 6850. ;
        c-gravity           = %point-552 ;
        diameter            = 115. ;
        slope-%             = 68.  ;
        is-in               = tende ;
        contains            = tr2 tr93 ;
        boundary-points     = po4 po6 po5 po1 ;
        connected-panels    = pp2 pp3 ;
        borders             = %border-589 %border-590 ;
        close-ridges        = ar1 ar3 ar4 ar5 ;
        above               = pp3 }
```

Reasoning maintenance is interesting in an interactive environment for spatial reasoning. As a matter of fact, the caching of inferences is necessary because of the size of the spatial knowledge base and the amount of inferences. In ELSA, an avalanche path can easily contain more than 500 triangles and 50 small panels and ridges. Without caching the time taken for the inferences will forbid any interactive use of the system, while ELSA is dedicated to decision support and thus needs interactive use.

But, in this kind of context, the user is also supposed to modify given knowledge. In ELSA, the user can change the vegetation of a part of a small panel (in order to simulate protection works for instance), or modify the definition of a small panel (toward a more accurate decomposition of space). As a result, the spatial properties of these small panels must be re-inferred. In order to keep the base consistent, a RMS is necessary.

Although ELSA is based on Shirka/TMS, it can take advantage of the RMS in order to manage dynamicity in spatial reasoning. Fig. 2 gives a good example of interest of such a RMS.

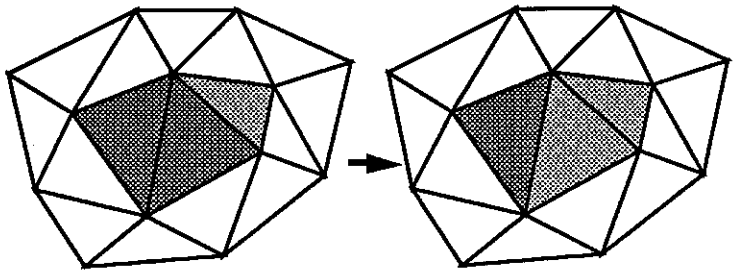

Fig. 2. In a triangulation of space, two polygons are defined through the set of triangles which are included in them. The inferences described below are made on those polygons. If a triangle changes its owner, the RMS must invalidate the cached inferences which were concerned by these two polygons. Meanwhile, the inferences conducted on the other polygons are not modified. The invalidation remains local.

As a summary, it appears that spatial reasoning applications can take advantage of classical RMS abilities. More precisely, the spatial locality can be translated in the dependency graph.

2.5. PERFORMANCE TESTS ON THE ELSA SYSTEM

Some inference times are given in order to illustrate our claims. They show how the caching is attractive and also why the RMS is useful. The tests have been performed on the same hardware as above.

Table 1. This first set of queries concerns caching; each query requires the computation of the close ridges of a panel. This second set of queries also concerns caching but queries compute the set of panels connected to a precise panel. No results about Shirka alone are provided because response times are prohibitive (in fact, from this unique test, we can conclude that ELSA is not viable without caching).

maintenance level	Shirka	Caching	RMS
Shirka: val? pp34 close-ridges	8.78	4.21	4.77
Shirka: val? pp34 close-ridges	8.72	0.	0.
Shirka: val? pp33 close-ridges	17.12	1.8	2.21
Shirka: val? pp32 close-ridges	19.01	2.11	2.49
Shirka: val? pp1 close-ridges	12.14	1.47	1.74
Shirka: val? pp2 close-ridges	8.71	1.14	1.36
Total 1	74.48	10.73	12.57
Shirka: val? pp26 connected-panels		65.32	75.92
Shirka: val? pp26 connected-panels		0.	0.
Shirka: val? pp27 connected-panels		4.9	6.7
Shirka: val? pp27 connected-panels		0.	0.
Shirka: val? pp30 connected-panels		4.15	5.37
Shirka: val? pp1 connected-panels		3.68	4.35
Shirka: val? pp2 connected-panels		3.53	4.96
Total 2		81.58	97.3

Table 2. After the tests that produced Table 1, the user changes the terrain description transferring one triangle (tr78) from a panel to another (just as in Fig. 2). The former queries are processed at new. In the first case (single caching), the user must clear the base and load it again. The time required for those operations is not taken into account.

maintenance level	Caching	RMS
Shirka: sup-val pp26 contains tr78	0.89	0.71
Shirka: aj-val pp27 contains tr78	0.1	1.81
Shirka: val? pp34 close-ridges	4.2	0.
Shirka: val? pp33 close-ridges	1.81	0.
Shirka: val? pp31 close-ridges	2.14	0.83
Shirka: val? pp1 close-ridges	1.49	0.83
Shirka: val? pp2 close-ridges	1.16	0.83
Shirka: val? pp26 connected-panels	65.87	6.16
Shirka: val? pp27 connected-panels	5.17	8.08
Shirka: val? pp30 connected-panels	4.17	7.
Shirka: val? pp1 connected-panels	3.7	5.17
Shirka: val? pp2 connected-panels	3.52	5.49
Total (initial inference + modification + re-inference)	175.8	134.21

With single caching, inference time is considerably reduced. A further discussion will give some explanations of some surprising results (especially the reduction of the *first* inference time). With the RMS, inference times are slightly increased in comparison with single caching inference times but the gain toward Shirka is obvious.

The second kind of queries shows the gain of time thanks to reasoning maintenance system. The comparison is made between single caching and RMS. The total line in Table 2 shows that the gain provided by the RMS is very important.

2.6. NEW EXPLANATIONS FOR THESE RESULTS: CONE EFFECTS

The observation made (comparing ELSA with or without RMS) are counter-intuitive at first sight:

1) Of course, the second call to the same inference takes no time with the RMS while, in spite of its the filtering capabilities, in Shirka, it still takes a while.
2) Even the first call is faster with the RMS than without (with a factor ≥2)!
3) Moreover, the time required to answer the same query against another object is reduced of a factor 8.

So these evaluations reveal a synergistic effect between inferences. These effects can be summarized as:

Backward cone effect: there is a backward cone effect when a datum is used several times in the computation of another. This can be stated in another way: the more used the datum, the better the caching. This effect is as much interesting as the datum is expensive to compute. Backward cone effect is able to explain the results above for points (2) and (3). Intermediate inferences performed use each other several times in order to obtain the high-level (or requested) data. With the RMS, these intermediate data are computed only once. For the same reason that the inferences of different data share the same intermediate inferences, after the computation of an item, the required time to answer the same query against another object is reduced. The two former points explain why the system is also faster on the re-computation after a change.

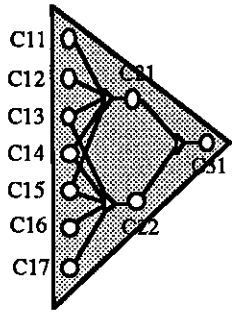

Fig. 3. In order to obtain C31, the system must infer C21 and C22 which in turn necessitates other inferences. Their computation can take advantage of caching because they share common inferences. This explains that the inferences produced with caching are faster even for their first computation.

Forward cone effect: the more used the datum, the worse the invalidation. As before, there is a forward cone effect when a datum is used for the inference of a important number of other pieces of knowledge. The forward cone effect is a negative effect, it reflects the necessary work in order to invalidate a cached result. It explains the classical results of observation (1) with Shirka/TMS.

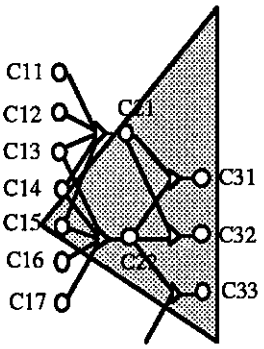

Fig. 4. The whole graph represents the inferences made by the inference engine. The shaded part of the graph is invalidated after the suppression of C15. We can see, *qualitatively*, that this shaded part looks like a "forward cone". The larger is this cone, the less interesting is the RMS because the number of inferences to launch is nearer from the numbers of all the inferences.

The problem that will be addressed in the remaining is: how is it possible to quantify these effects? and which conclusions to draw for the use of a RMS in a particular application. It is obvious that the attraction of a RMS in an application will result in a trade off between backward and forward cone effects.

3. A SPACE OF REASONING: THE DEPENDENCY GRAPH

Here is an attempt to generalize the results we obtained with the ELSA experiments in order to state what kind of reasoning/application can benefit from a RMS.

Real efficiency of RMS is very difficult to evaluate because a lot of factors have to be taken into account: not only the number of nodes and justifications but also the way they are organized in cycles of different kind and the order of firing rules. Moreover, the performances of RMS depend heavily of the kind of use. Here, we do not address these complexity problems but the conditions under which a RMS is useful in order to maintain a reasoning. So, worst case analysis is not a suited measure of the performances of the system and an abstract computation of the algorithm complexity is not very useful. What is important for real applications is not the theoretical complexity analysis of the program used for reasoning maintenance but the real complexity of the RMS when confronted with the real reasoning. To that extent, we exhibit some results for graphs with particular restrictions that do not trigger the whole machinery of a RMS.

This section will, first, set some definitions to be used in the quantitative analysis and the restrictions used in the present study. Then the analysis is achieved for both kinds of cone effect before summarizing the results of the tradeoff between backward and forward cone effect.

3.1. NOTATION AND RESTRICTION

In order to give some precise results, some hypotheses have been done about the dependency graph. We assume that:

H1) There is no nonmonotonic inferences. This is not an important restriction when assumed the second hypothesis. In fact, nonmonotonic inference in a graph without loops is a problem for the inference system but not for the RMS.

H2) There is no loops in the graph. This assumption is quite restrictive. In fact, it is restrictive regarding the complexity analysis of RMS, but it is not for a lot of applications.

H3) The analysis below only considers average values and hypothesizes the homogeneity of the graph. With regards to real application, this is the most restrictive hypothesis. The general aspect of reasoning will be evaluated and quantified on the basis of average values considered that the graph can itself be decomposed in several little sub-graphs in which it is possible to cancel or activate reasoning maintenance.

All those hypotheses are set for reason of simplicity. Of course, the quantitative analysis of reasoning for RMS have to be fulfilled with the relaxation of those hypotheses.

First, some notations have to be introduced. Let B be a knowledge base dedicated to a given application. We consider all the inferences launched all along the typical session of the application; this is called the reasoning. A particular reasoning can be represented as a dependency graph such as the one used in the RMS. If we do not care for nonmonotonic inferences (H1), it is an AND-OR graph (each inference is an and-node linking the antecedents to the consequent, each formula is an or-node linking together the possible inference of this formula).

Note that the dependency graph (as it does in RMS) does not represent the potential inference of B, but the inferences really committed. The formulas in the graph constitute the set F of formulas used in the reasoning (they can either be given by the user or inferred by the reasoning system). N is the number of all formulas in F. In F, we distinguish two sets of formulas: I is the set of initial formulas which are given and not inferred, and Q is the set of interesting formulas which are the goal of the reasoning process.

We call a chain, a sequence $f_0, j_1, f_1, \ldots j_n, f_n$ of formulas and justifications such that, for each $i \in [1,n]$, f_{i-1} is an antecedent of j_i and f_i is the consequent of j_i in the graph. n is the length of the chain (the number of justifications).

The forward depth (df(f)) at node f is the length of the longest chain beginning at node f (and ending at a node in Q). The backward depth (db(f)) at node f is the length of the longest chain ending by node f (and beginning at a node in I). Backward depth is also called the level of f.

The forward width (wf(f)) at node f is the number of and-node f is linked with as antecedent. The backward width (wb(f)) at node f is the number of and-node f is linked

with as consequent. So, wf(f) and wb(f) are the number of connections at front or back of an OR-node. Note that width can also be called branching factor at node f.

wf is the average number of justifications based on one formula of F\Q (where F\Q={x; $x \in F \wedge x \notin Q$}). wb is the average number of justifications of a formula of F\I. In this paper, we will consider that wb = 1 (this means that a datum is inferred by only one way). So,

$$wf = \frac{\sum_{f \in F \setminus Q} wf(f)}{|F \setminus Q|}, \qquad wb = \frac{\sum_{f \in F \setminus I} wb(f)}{|F \setminus I|}$$

$\nu(f)$ is the number of times f is used during the session, this is not the number of inferences in which it appears but the number of times these inferences are drawn. In a lot of applications these inferences are used a lot.

ρ is the ratio $\frac{|F \setminus Q| * wf}{|F \setminus I| * wb}$. Thus, here $\rho = \frac{|F \setminus Q| * wf}{|F \setminus I|}$, because of the value of wb. It is the average number of antecedents per justification in the reasoning graph.

We distinguish several constant times which are:

τ_{inf}: the average time taken for an inference for which all the premisses are available.
τ_{rec}: the average time taken for recording a value (result of an inference).
τ_{dep}: the average time taken for recording a dependency (representing an inference).
τ_{sup}: the average time taken for suppressing a dependency and a cached value.

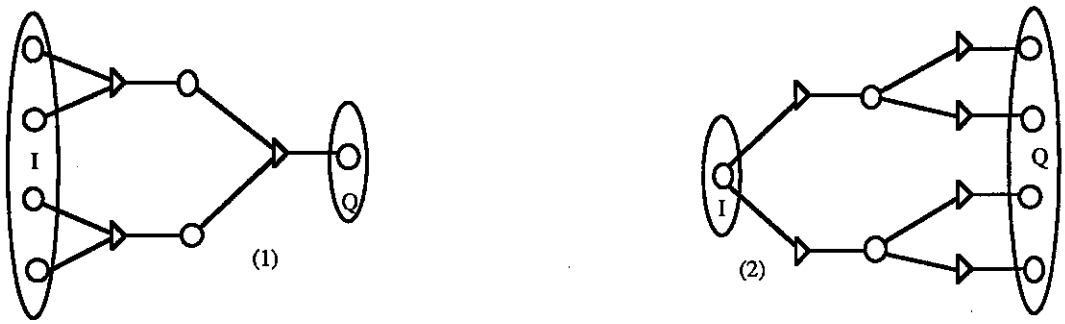

Fig. 5. Examples of typical graphs

An additional important constant time also appears, but is not taken into account in our argumentation. It is T_{reset}, the time required for quitting and loading the application again. We guess that all these values can be easily evaluated for homogeneous reasoning. Two archetypical examples are given in Fig. 5.

Table 3. The average values of the variables for the graphs given above.

	|F|=N	|Q|	|I|	wf	wb	ρ
(1)	7	1	4	1	1	2
(2)	7	4	1	2	1	1

3.2. INFERENCE AND PROPAGATION ANALYSIS

At first sight, the time taken in order to produce the reasoning is

$$TB = \tau_{inf} * \sum_{f \in FI} \nu(f)$$

But, if the backward cone effect is accounted for, we can say (if $\rho \neq 1$) that the time taken to infer the formula f is:

$$TB(f) = \tau_{inf} * \frac{\rho^{db(f)} - 1}{\rho - 1}$$

because $\frac{\rho^{db(f)} - 1}{\rho - 1}$ is the size of a complete ρ-ary tree of depth db(f). If $\rho = 1$, then $TB(f) = \tau_{inf} * db(f)$. Hence, the total time for the inferences of the session, if no result of inference is recorded (and $\rho \neq 1$), is:

$$TB = \tau_{inf} * \sum_{f \in Q} \frac{\rho^{db(f)} - 1}{\rho - 1}$$

It is noteworthy that the ratio used in TB(f) is the number of inferences in the backward cone. If other hypotheses are taken into account (no homogeneity, no tree structure...), the formula can be replaced in TB(f) by another expressing the number of inferences in the cone. With the recording of all the inferred formulas the time is:

$$TB_{cach} = \sum_{f \in FI} (\tau_{inf} + \tau_{rec}) = |FI| * (\tau_{inf} + \tau_{rec})$$

and

$$TB_{RMS} = \sum_{f \in FI} (\tau_{inf} + \tau_{rec} + \tau_{dep}) = |FI| * (\tau_{inf} + \tau_{rec} + \tau_{dep})$$

As a result, the gain for RMS is:

$$GB = \sum_{f \in FI} [(\nu(f) - 1) * \tau_{inf} - (\tau_{rec} + \tau_{dep})]$$

the formula is the same for caching without the reference to τ_{dep}.

Table 4. The values for the graphs given as examples. Notes that they are not multiplied by the same factors. As a result, case (1) do not profit from caching (and this is true whatever is the total depth of the graph).

	TB_1	TB_2	TB_{cach}	TB_{RMS}	GB
(1)	3	3	3	3	0
(2)	8	8	6	6	2

3.3. INVALIDATION ANALYSIS

A RMS is useful for invalidation (otherwise, rough caching is enough as it is in forward chaining systems). Invalidation will lead to the forward cone effect. This effect is now evaluated. We consider that one given formula f in I is modified and that the user asks the same queries Q as before. Because of the caching system, the answers

recorded are no valid any more. With a single caching system, we need to re-infer all the formulas. The time required is so

$$TF_{cach} = TB_{cach} + T_{reset} + TB_{cach}$$

where $TB_{cach} = |F\backslash I| * (\tau_{inf}+\tau_{rec})$ as above.

Another solution uses a reasoning maintenance system. In that case,

$$TF_{RMS} = TB_{RMS} + T_{inv} + T_{reinf}$$

with $TB_{RMS} = |F\backslash I| * (\tau_{inf}+\tau_{rec}+\tau_{dep})$ as above,

and $T_{inv} = N_{inval}*\tau_{sup}$ and $T_{reinf} = N_{inval}*(\tau_{inf}+\tau_{rec}+\tau_{dep})$,

and $N_{inval} = \frac{wf^{df(f)+1} - 1}{wf - 1} - 1$ if $wf \neq 1$ and $df(f)$ otherwise, this is, again, the size of a wf-ary tree of depth df(f), so this is the size of the forward cone starting at f. The branching factor is wf because it has been considered justifications with only one consequent, otherwise, the branching factor would have been wf*nbcsq (in which nbcsq is the average number of consequents).

As a result, the gain given by the RMS is:

$$GF = N*(\tau_{inf}+\tau_{rec}) + T_{reset} - N_{inval}*(\tau_{inf}+\tau_{rec}+\tau_{dep}+\tau_{sup})$$

If we ignore resetting time and set that $(\tau_{inf}+\tau_{rec})*k = \tau_{inf}+\tau_{rec}+\tau_{dep}+\tau_{sup}$, the RMS must be attractive when $N_{inval}*k < N$ which must be true most of the time.

Table 5. The results of the invalidation phase for the graphs given above ($\tau_{cach}=\tau_{inf}+\tau_{rec}$ and $\tau_{RMS}=\tau_{inf}+\tau_{rec}+\tau_{dep}+\tau_{sup}$). The graphs are not big enough to illustrate interesting properties: both cases do not appeal for a RMS. In particular, locality do not appear (after each modification, an important part of the graph must be revised). It becomes more attractive if we consider 10 independent graphs as in case 2' in which the invalidation is useful.

	TF_{cach}	N_{inval}	TF_{RMS}
(1)	$6*\tau_{cach}$	2	$6*\tau_{RMS}+2*\tau_{sup}$
(2)	$12*\tau_{cach}$	6	$12*\tau_{RMS}+6*\tau_{sup}$
(2')	$120*\tau_{cach}$	6	$66*\tau_{RMS}+6*\tau_{sup}$

A pure static evaluation can be given with p modifications of data and the whole set of queries between them:

$$T(p) = p * TB$$

$$T_{cach}(p) = p * T_{reset} + (p+1) * TB_{cach}$$

$$T_{RMS}(p) = TB_{RMS} + N_{inval} * p * (\tau_{inf}+\tau_{rec}+\tau_{dep}+\tau_{sup})$$

3.4. REASONING (ABOUT/FROM) THE GRAPH

As said above, the main problem consists in evaluating the tradeoff between both cone effects. The important questions to ask for a particular application are: Can it benefit from caching ? Does caching need dynamicity management ? Is a RMS suited for dynamicity management or is it better to recompute everything ?

It is noteworthy that these questions cannot be answered independently. Moreover, they are not directive; in particular, dynamicity management does not imply the use of a RMS. Nevertheless, reasoning dynamics must be taken into account. As a matter of fact, the performances of the system depend on the relations between query and modification time. The result will not be the same if there is a new query after each modification or if there is an important number of modifications between each query phase.

4. CONCLUSION

The problem we addressed was the evaluation of the benefits of caching and RMS in knowledge based applications. To that extent, we first show some results expected on a general purpose tool and some results obtained on a real world application. The results, at the advantage of the RMS, were not expected. We explained then by producing two informal models of the actions of caching and RMS on the reasoning: the so-called cone effects. Then, we quantified the amount of work required in order to demonstrate some facts (or resolve one problem). The equations we obtained revealed the presence of the cone in the quantifications of the number of inferences they contain.

This is a first attempt in order to characterize the usefulness of caching and RMS. It has to be continued by a better knowledge of reasoning dynamics and by relaxing the hypotheses we assumed on the graphs. Finally, the advantage of using a RMS in an application is seen as a tradeoff between both principles. More examples and experiments, together with a discussion of further and related works can be found in [5].

REFERENCES

1. F. Rechenmann and P. Uvietta, "SHIRKA: an object-centered knowledge base management system", *Artificial intelligence in numerical and symbolic simulation*, 9-23 (1991).

2. J. Euzenat, "Cache consistency in large object knowledge bases", Laboratoire ARTEMIS/Imag internal report (1990).

3. L. Buisson, "ELSA : a problem solving environment for avalanche path analysis", *Artificial intelligence in numerical and symbolic simulation*, 25-49 (1991).

4. L. Buisson, "Reasoning on space with knowledge object centered representation", *Lecture notes on computer science*, 409:325-344 (1990).

5. L. Buisson and J. Euzenat, "A quantitative analysis of reasoning for RMSes", Laboratoire ARTEMIS/Imag Internal report (1991).