# FORMAL REAL-TIME IMAGINATION*

MADHURA NIRKHE
Department of Electrical Engineering
University of Maryland, College Park, MD 20742
madhura@cs.umd.edu

SARIT KRAUS
Department of Mathematics and Computer Science
Bar Ilan University
Ramat Gan, 52900 Israel
sarit@bimacs.cs.biu.ac.il

Note: This article appears in *Fundamenta Informaticae*, special issue on Context: theory and practice, 23(2,3,4):371–390, 1995.

**Abstract**

Formal real-time imagination is a term that may curiously describe the activities of a commonsense agent in a real-time setting in general, and in a tight deadline situation in particular. We briefly describe an 'active-logic' mechanism that fits this description. Temporal projection is an essential component of real-time planning. We draw a parallel between imagination as we understand it in human context and the capacity of the automated agent to formulate mental images of possible scenarios and plans of action in the course of its reasoning. We outline a treatment of temporal issues of significance to a time-situated reasoning mechanism in a dynamic setting with deadlines. The Yale shooting problem is a benchmark problem in temporal reasoning. We demonstrate how the active-logic planning mechanism successfully handles some interesting real-time variants of the Yale shooting problem. The solutions to each of these illustrate the agent's ability to form *contexts* within which to reason, to *project* in each context thus formed by applying default inferences, and to *revise* and extend its conclusions within each context by applying time-sensitive inference rules, and most importantly, to account for *all* the time spent in the process.

*This work was supported in part by NSF grants No. IRI-9123460 and IRI-9311988. The second author is also affiliated with the Institute for Advanced Computer Studies, University of Maryland, College Park.

# 1 Introduction

Imagination is not a quality that one normally tends to attribute to robots. The Webster dictionary defines imagination as:

Def. 1: the act or power of forming a mental image of something not present to the senses or never before wholly perceived in reality
Def. 2a: creative ability
Def. 2b: ability to confront and deal with a problem: resourcefulness
Def. 2c: the thinking or active mind
Def. 3a: a creation of the mind; esp: an idealized or poetic creation
Def. 3b: fanciful or empty assumption

While the term 'formal imagination' may outwardly seem like an oxymoron, a careful examination reveals attributes crucial to automated agents that are not too distant from the definitions above. We have formulated a mechanism for *fully* real-time deadline coupled planning using *active logics* [13, 21]. Uncertainty, urgency and unknowns characterize the complex world of a commonsense agent. Dudley, our agent, is in a particularly tight setting with respect to deadlines. He must formulate and enact a plan to save Nell who is tied to the railroad tracks as a train approaches in the distance[1]. Formulating plans that are deadline feasible certainly involves the power to form a mental image of something not perceived before (Def. 1). Dudley must make *creative* use of his limited resources and exhibit an ability to confront and deal resourcefully with his problem (Def. 2b). Default assumptions must be made to envision a state of the world that will result from successful execution of his actions. These are indeed fanciful or empty assumptions (Def. 3b) that he must revise nonmonotonically in the face of evidence to the contrary. The underlying mechanism is a formal active logic [6, 24]. In this sense, Dudley's 'active mind' can use its formal declarative machinery to take leaps of imagination and the term 'formal real-time imagination' could aptly describe Dudley's behavior in real-time using active-logics.

Temporal projection is an essential component of real-time planning. A parallel can be drawn between imagination as we understand it in human context and the capacity of the automated agent to formulate mental images of possible scenarios and plans of action in the course of its reasoning. We outline a treatment of temporal issues of significance to a time-situated reasoning mechanism in a dynamic setting with deadlines. The Yale shooting problem is a benchmark problem in temporal reasoning. We demonstrate how the active-logic planning mechanism successfully handles some interesting real-time variants of the Yale shooting problem. The solutions to each of these illustrate the agent's ability to form *contexts* within which to reason, to *project* in each context thus formed by applying default inferences, and to *revise* and extend its conclusions within each context by applying time-sensitive inference rules, and most importantly, to account for *all* the time spent in the process.

The frame problem in AI [16] is the problem of deciding what persists and what changes as actions are performed. Early STRIPS style planners in static settings explicitly modeled the effects of actions in the form of add and delete lists to get around

[1]This problem was first mentioned in the context of time-dependent reasoning by McDermott [17], and more recently discussed in [3].

$$
\begin{array}{llcl}
0: & & \emptyset & \\
\vdots & & & \\
i: & \dots & \alpha & \dots \\
i+1: & \dots & \alpha \rightarrow \beta, \beta \rightarrow \gamma & \dots \\
i+2: & \dots & \beta & \dots \\
i+3: & \dots & \gamma & \dots \\
\vdots & & &
\end{array}
$$

Figure 1: Step-logic studies

the persistence problem. The world does not change much here, a single model is updated to reflect the results of actions. Two other related frame issues have been well documented in the literature [9]: The qualification problem is the impossibility of accounting for an arbitrarily large number of potential (pre)conditions for an action and the ramification problem is the difficulty in reasoning about the extended effects of actions as time passes. In this paper we also show how Dudley's fully deadline-coupled reasoning mechanism offers a real-time solution to the above aspects of the frame problem.

We have formulated a combination of declarative and procedural approaches to model Dudley. The underlying mechanism is that of Elgot-Drapkin's step-logics (which have since been extended and renamed as active logics) which are described in section 2. In section 3 we sketch the temporal reasoning inference rules for frame-default reasoning. These form part of the real-time planning infrastructure. As a litmus test, we describe in section 4 how the same agent Dudley would behave in various dynamic versions of the infamous Yale shooting problem [10, 7, 8, 12, 15, 20, 23, 25, 2]. Section 5 describes Dudley's real-time treatment of the qualification and ramification problems. Section 6 compares the active logic approach to other research in temporal reasoning.

## 2 One Step At a Time

Step-logics [6] were introduced to model a common sense agent's ongoing process of reasoning in a changing world. A step-logic is characterized by a language, observations and inference rules. A *step* is defined as a fundamental unit of inference time. Beliefs are parameterized by the time taken for their inference, and these time parameters can themselves play a role in the specification of the inference rules and axioms. The most obvious way time parameters can enter is via the expression $Now(i)$, indicating the time is now $i$. Observations are inputs from the external world, and may arise at any step $i$. When an observation appears, it is considered a belief in the same time-step. Each step of reasoning advances $i$ by 1. At each new step $i$, the only information available to the agent upon which to base his further reasoning is a snap-shot of his deduction process completed up to and including step $i-1$.

The agent's world knowledge is in the form of a database of beliefs. These contain domain specific axioms. A number of inference rules constitute the inference engine. Among them may be rules such as Modus Ponens and rules to incorporate new observations into the knowledge base as well as rules specific to deadline-coupled planning such as checking the feasibility of a partial plan or refining a partial plan. Figure 1, adapted from [5] illustrates three steps in a step-logic with Modus Ponens ($\frac{i:\alpha,\alpha\rightarrow\beta}{i+1:\beta}$) as one of its inference rules.

## 2.1 Inside the active-logic planner

We start by providing a brief description of the syntax used in the deadline-coupled planning mechanism based on active logics. A formula $X(S:F,Args)$ consists of a predicate name $X$ which may represent a fluent or an action predicate, with a list of arguments. The first argument denotes the time interval $S:F$ over which the predicate holds and $S$ and $F$ are the end points of the interval. The other arguments of the predicate follow and are denoted by $Args$ for easy reference. We often wish to express formulas with predicates that hold only over the duration of their interval $S:F$ and do not continue to hold beyond $F$ by persistence. Most of the predicates denoting agent actions (e.g., *Run*, *Shoot*) fall in this category. We denote the time intervals in formulas involving these predicates by $S:\overline{F}$. We use the shorthand $S$ for $S:S$ to denote an instantaneous action over a point time interval.

Further, we have four different forms of formulas. $X(S:T,Args)$ denotes that $X$ holds over interval $S:T$, $\neg X(S:T,Args)$ denotes that $\neg X$ holds over $S:T$. Further, $X_c(S:T,Args)$ (resp., $\neg X_c(S:T,Args)$) are used to denote that not only does the agent believe in $X$ (resp., $\neg X$) over the interval, but that the agent has reason to believe that the time point $S$ could be a possible point of change of the predicate from $\neg X$ to $X$ (resp., from $X$ to $\neg X$) in the event that $\neg X$ (resp., $X$) holds in the interval ending in $S$. For example, $On(2:4,floor,ball)$ should be read as the ball was on the floor in the interval $2:4$ and $On_c(2:4,floor,ball)$ is read as the ball was on the floor over the interval $2:4$ and probably wasn't there before 2.

Either of $\{X(S:F,Args),X_c(S:F,Args)\}$ are defined to be in *direct contradiction* with either of $\{\neg X(S:F,Args),\neg X_c(S:F,Args)\}$. A *uniqueness contradiction* exists between formulas $X(S:F,Args1,U,Args2)$ and $X(S:F,Args1,V,Args2)$ if $X(S:F,Args1,U,Args2)\rightarrow\neg X(S:F,Args1,V,Args2)$ whenever $U\neq V$. E.g., there is a uniqueness contradiction between $At(5,Dudley,home)$ and $At(5,Dudley,railroad)$. A formula $\alpha$ *du-contradicts* a formula $\delta$, if it is in direct or uniqueness contradiction with $\delta$. The same definitions of contradiction extend to $X_c$ and the negated versions.

We use annotated formulas such as $X(S:F,Args)[\beta_1,\ldots,\beta_k]$ to denote a formula that is derived using the default formulas (projections) $\beta_1,\ldots,\beta_k$ in its proof. Such an annotated formula itself has the status of a default and is as feasible as the weakest default in the annotation as explained in Sections 3.1 and 3.3.

An action triplet denoted by $[C_A,A,R_A]$ consists of an *action* $A$ preceded and followed, respectively, by a list of *conditions* $C_A$ and *results* $R_A$. $A$ is a formula with an action predicate and $C_A$ and $R_A$ are lists of formulas[2]. The conditions may need to be true over all or some of the duration of the action. An action may be complex or primitive (atomic). A primitive action takes one time step to perform. Firing of an

[2]Whenever formulas appear in lists such in $C_A$ or $R_A$ and later in beliefs **CS**, **Proj** and **Ppl**, they are in fact treated as if they are "quoted". We omit the quotes to keep the long strings readable. Thus the beliefs of the agent that we will describe shortly are still first order formulas.

inference rule corresponds to a *think* action. Dudley's non-defeasible beliefs are treated as *facts*[3]. Observations incorporate into beliefs in the same time step. Theorems whose premises consist of facts alone are also regarded as facts.

$Now(i)$ denotes Dudley's belief that the time is currently $i$. A partial plan is a belief $\mathbf{Ppl}(i, p, Triplet\_List)$ denoting a partial plan at step $i$ with the name $p$. The $Triplet\_List$ is an ordered list of action triplets. We will sometimes use $\mathbf{Ppl_{i,p}}$ to denote the $Triplet\_List$ with respect to $i$ and $p$.

A special plan with the name *null* is a plan with no actions in it. Dudley simultaneously develops alternative plans towards attaining his goals or subgoals. Each of these partial plans (including the null plan) defines a context within which reasoning can be done about the expected state of the world if the plan were to be carried to completion.

The agent maintains a belief $\mathbf{CS}(i, p, Context\_List)$ denoting the **Context_set** for each plan $p$ at each step $i$. The list $Context\_List$ consists of quoted formulas (we omit the quotes for readability), and includes all of the facts (observations)[4], formulas corresponding to actions in the plan and formulas that the agent deduces to be true in the state of the world resulting from the successful execution of plan $p$. We will often use $\mathbf{CS_{i,p}}$ to denote the list $Context\_List$. All formulas corresponding to a given predicate $X$ are kept sorted in the list with $S : F$ of each formula as the key. The context set changes with time as the plan undergoes modification and as inferences are made in the context of the plan.

At each step $i$, the belief $\mathbf{Proj}(i, p, Proj\_List)$ denotes the projection that is formed in the context of each partial plan $p$ that is in progress, based on the default of persistence[5]. The $i$ denotes the step number, and $Proj\_List$ is a list of quoted formulas. We will often use $\mathbf{Proj_{i,p}}$ to denote the list $Proj\_List$ with respect to $i$ and $p$.

The belief $\mathbf{Goal}(p, G, D)$ is maintained to denote that the plan $p$ is being developed to meet a goal $G$ by a deadline $D$.

# 3 Frame-default Inferencing

This section describes Dudley's inference mechanism for temporal reasoning. We describe three inference rules that are crucial for this: temporal projection rule (TP), context set revision rule (CSR) and the restructured modus ponens rule (RMP). In all the active logic scenarios we have underlined <u>new</u> formulas in the context sets or projections, to highlight the differences with the corresponding beliefs at the previous step. In each step the TP rule derives a new projection in the current context and the RMP and CSR rules together deduce a new context set.

[3]Strictly speaking though, the agent only has beliefs, never facts, since even observations are not etched in stone, and may very well change over time. In all the problems that we tackle though, we will treat observations and facts synonymously.

[4]Actually it only consists of the subset of facts that is relevant to the particular partial plan. [22] deals with space bounds on the reasoning and proposes a relevance mechanism to keep the reasoning directed to a particular partial plan for a duration of time.

[5]Projections (and persistences) have been studied by numerous authors; see e.g., [26, 12, 18]. Our treatment is along the lines of time-maps of [4].

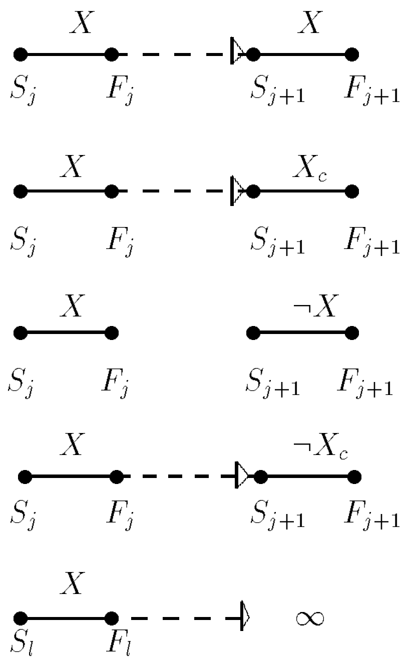


Figure 2: Pictorial description of the TP rule

## 3.1 Temporal projection rule (TP):

The temporal persistence rule (TP) effectively *smoothes* beliefs over time intervals which present gaps in the agent's knowledge. At each step, $\mathbf{Proj_{i,p}}$ holds the results of the temporal projection rule applied to the context set $\mathbf{CS_{i-1,p}}$ of the previous step. Our approach can be best described by a term which we call *parallel projection*. That is, the entire known state of the world at one moment is used to determine the (expected) state at the next moment. Since active-logics are built around the idea of specifying what is known (e.g., proven) *so far*, all predicates, and all context sets can be simultaneously reconsidered at each new time step.

Here is a description of the TP rule applied to atomic formulas corresponding to a given predicate $X$ in the context set at step $i$ in order to constitute $\mathbf{Proj_{i+1,p}}$ for the partial plan $p$. Note that formulas of the form $X(S:\overline{F}, Args)$ are not projected, since they are intended to represent mostly agent actions that do not persist. Formulas $X(S:F, Args)$ for each predicate $X$ are kept sorted in $\mathbf{CS_{i,p}}$ with $(S:F)$ as the key. Only predicates corresponding to *fluents* (i.e. those formulas without the bar on top as in $S:\overline{F}$) are eligible for projection. The formulas in $\mathbf{Proj_{i+1,p}}$ are strictly those that are obtained by persistence of those in $\mathbf{CS_{i,p}}$. Let $\alpha_j$ and $\alpha_{j+1}$ denote consecutive formulas in sorted order in $\mathbf{CS_{i,p}}$ and let $\alpha_l$ denote the last formula in this order. The TP rule can then be described as follows[6]: Figure 2 shows a pictorial description of the TP rule, with the dashed lines denoting the intervals that are filled with the projection.

1. If $\alpha_j$ is of the form $X(S_j:F_j, Args)$ and $\alpha_{j+1}$ is of the form $X(S_{j+1}:F_{j+1}, Args)$ then $\mathbf{Proj_{i+1,p}}$ contains $X(F_j+1:S_{j+1}-1, Args)$ whenever $F_j < S_{j+1}$.

[6]For brevity, we only describe the rule for $\alpha_j = X(S_j:F_j, Args)$. The same applies to $\alpha_j = X_c(S_j:F_j, Args)$. The dual form involving $\neg X$ is similar.

2. If $\alpha_j$ is of the form $X(S_j : F_j, Args)$ and $\alpha_{j+1}$ is of the form $X_c(S_{j+1} : F_{j+1}, Args)$ then $\mathbf{Proj_{i+1,p}}$ contains $X(F_j + 1 : S_{j+1} - 1, Args)$ whenever $F_j < S_{j+1}$.

3. If $\alpha_j$ is of the form $X(S_j : F_j, Args)$ and $\alpha_{j+1}$ is of the form $\neg X_c(S_{j+1} : F_{j+1}, Args)$ then $\mathbf{Proj_{i+1,p}}$ contains $X(F_j + 1 : S_{j+1} - 1, Args)$ whenever $F_j < S_{j+1}$.

4. If $\alpha_j$ is of the form $X(S_j : F_j, Args)$ and $\alpha_{j+1}$ is of the form $\neg X(S_{j+1} : F_{j+1}, Args)$ then $\mathbf{Proj_{i+1,p}}$ does not speculate over the truth or falsity of $X$ over $F_j+1 : S_{j+1}-1$. The projection rule will smooth over this interval when further information about a possible point of time where the value of $X$ changes becomes available.

5. If $\alpha_l$ (with the latest interval in $\mathbf{CS_{i,p}}$ corresponding to the predicate $X$ is of the form $X(S_l : F_l, Args)$ or $X_c(S_l : F_l, Args)$ then $\mathbf{Proj_{i+1,p}}$ contains $X(F_l + 1 : \infty, Args)$.

If any of $\alpha_j$ and $\alpha_{j+1}$ in the context set du-contradict, the projection is frozen for the interval in dispute until the contradiction is resolved. In case of a contradiction, it may sometimes be necessary to break up the formulas into two or more parts to identify the extent of the contradiction over some sub interval. e.g., when $X(5)[Y(4)]$ and $\neg X(1 : 6)$ are in a context set, the latter must be split into $\{\neg X(1 : 4), \neg X(5), \neg X(6)\}$ to identify the range of the contradiction.

In case of a contradiction in the context set, between two formulas, $\alpha_j$ and $\alpha_{j+1}$ in the sorted order, Dudley's TP rule can not decide on which of the contradicting formulas to project, until the contradiction is resolved. Often, one of the contradicting formulas is a fact while the other is weaker since it is only based on projections, and the contradiction is resolved in subsequent steps. The projection is frozen for subsequent $\alpha_{j+2}$, $\alpha_{j+3}$ , .... However, when a fact $\alpha_k = X(S_k : F_k, Args)$ is encountered down the chain, Dudley can resume projection starting with the interval $S_k : F_k$ by applying the above procedure to the remainder of the sorted list with $\alpha_k$ as its first element. Projections are not made in the interval between the contradiction and $S_k$, until the contradiction is resolved.

**Example**
Consider a scenario with a bucket filled with oil and a ball lying on the floor. This example illustrates an application of the TP rule to $\mathbf{CS_{15,p}}$ to yield $\mathbf{Proj_{16,p}}$.

$\mathbf{15} : \ldots, \mathbf{CS}(15, p, \{\ldots, filled(0), filled(4), \neg filled_c(7), filled(10),$
$on(0, floor, ball), \neg on_c(5, floor, ball), \ldots\}), \ldots$

$\mathbf{16} : \ldots, \mathbf{Proj}(16, p, \{\ldots, filled(1 : 3), filled(5 : 6), filled(11 : \infty),$
$on(1 : 4, floor, ball), \neg on(6 : \infty, floor, ball), \ldots\}), \ldots$

## 3.2 A restructured Modus Ponens (resolution) rule (RMP):

Instead of applying MP in its familiar form : viz. from $\alpha$ and $\alpha \rightarrow \beta$ deduce $\beta$, we choose a representation in clause form and apply a restructured MP in accordance with our philosophy to let earlier defaults play out their effects completely to result in an anticipated state of the world to which later defaults may be applied if necessary. A formula which is a fact has no justification attached to it. All axioms are treated as facts. A formula $\alpha$ which was derived using one or more projections $\beta_1, \beta_2 \ldots$ is only as feasible as the weakest projection, and is itself classified as a default. Such a formula is annotated with the projections used in its derivation and is written as $\alpha[\beta_1, \beta_2, ...]$.

Let $\neg\alpha_1 \vee \neg\alpha_2 \vee \ldots \vee \neg\alpha_n \vee \beta$ be the expression in clause form that appears in the context set $\mathbf{CS_{i,p}}$ of a plan $p$ at step $i$. This may either be an axiom or an observation. We formulate a rule that adds new atomic formulas (with or without justifications) derived within a context to $\mathbf{CS_{i+1,p}}$. We use the terms, *finished* and *unfinished* to describe a resolution where the results are atomic and non-atomic formulas respectively. The rule only carries over the result of a finished resolution to the context set at the next step[7].

The process of resolution can be outlined as follows:

- All the $\alpha_j$ from $\mathbf{CS_{i,p}}$ which are facts are first used to resolve. If there are no facts that are eligible for resolution, the resolution is not carried out at all. There must be at least one fact among the resolvents for the RMP to fire[8].
- Subsequently, if the resolution is unfinished, members from $\mathbf{CS_{i,p}} \cup \mathbf{Proj_{i,p}}$ which are themselves defaults are next tried. All formulas from $\mathbf{Proj_{i,p}}$ as well as those formulas in $\mathbf{CS_{i,p}}$ that are annotated with projections qualify as defaults. From among those in $\mathbf{Proj_{i,p}}$, those with earlier time parameters are used before the ones with later parameters. For the annotated formulas, the annotation with the latest time parameter that was used in the derivation is used to decide the priority[9].
- The result of the resolution $\beta$ is then annotated with all the projections used, either directly, or in the annotation of resolving formulas from the context set. The annotations are attached in square brackets to the formulas. This provides the basis for a real-time truth maintenance mechanism which is useful in resolving contradictions.
- If a projection $\alpha$ with a later time than the time of $\beta$ is used in the RMP application, $\beta$ is not added to the context set. It is discarded. Thus the axioms are used to derive future conjectures based on projections of current beliefs, but prevented from using future projections to derive past conclusions and from jumping to extreme conclusions as we demonstrate in Section 4.

[7]Since we have the luxury of applying all rules to all formulas at every step, not much is to be gained by adding the results of an unfinished resolution to the context set. We wait to get more information later such as from observations or from further deductions so that an atomic formula can be derived. This serves the purpose of limiting the size of the context set. It is possible to write a version of the RMP that will also add the results of unfinished formulas to the context set, and would effectively function the same but use more space.

[8]The motivation behind this is to reduce the large number of formulas resulting from applying RMP to projections alone, since what can be derived thus is also obtained by the combined effect of RMP and TP. This reduces the actual number of formulas in the context set without loss of any meaningful commonsense conclusions. As an example, consider the axioms $Alive(T) \rightarrow \neg Dead(T)$. Suppose $Alive(0)$ is the only formula in the context set. By TP, the agent would add $Alive(1:\infty)$ to the projection. and by RMP, $\neg Dead(0)$ to the context-set. Note that $\neg Dead(1:\infty)$ will subsequently be in the projection, and there is no need to additionally have $\neg Dead(1:\infty)[Alive(1:\infty)]$ in the context set. Hence this is a reasonable way to curtail the size of the context set.

[9]In case of a tie, we draw all the conclusions resulting from the use of the projections with identical time intervals, one at a time. This may result in implicit contradictions. In this situation, the system deduces an expected contradiction.

- If $\beta = X(S : F, \ldots)$ has its time interval $S : F$ such that $S$ is later than the time intervals of all the $\alpha_j$ used in the resolution, then it is marked as $X_c(S : F, \ldots)$ in the context set, to denote that it could be a potential point of inflection in the value of the predicate $X$.[10] This marking is of help in deciding whether to project $X$ such as in the TP rule above.

**Example**
This example illustrates two applications of RMP (in steps 4 – 6) following Dudley's observation of someone dropping a ball into a full bucket. Given the axioms that a ball dropped into a full bucket results in a spill, Dudley concludes that the floor will no longer be dry[11].

**Axioms** (These are part of the *Context_List* of every context set):
$\neg Filled(T) \vee \neg Drop(\overline{T}, ball) \vee Spill(\overline{T+1})$
$\neg Spill(\overline{T}) \vee \neg Dry(T+1, floor)$

$$\mathbf{4} : \mathbf{CS}(4, null, \{\ldots, Dry(0, floor), Filled(0), \underline{Drop(\overline{4}, ball)}\}),$$
$$\mathbf{Proj}(4, null, \{\ldots, Dry(1:\infty, floor), Filled(1:\infty)\})$$

$$\mathbf{5} : \mathbf{CS}(5, null, \{\ldots, \underline{Spill_c(\overline{5})[Filled(4)]}, Dry(0, floor), Filled(0), Drop(\overline{4}, ball)\}),$$
$$\mathbf{Proj}(5, null, \{\ldots, Dry(1:\infty, floor), Filled(1:\infty)\})$$

$$\mathbf{6} : \mathbf{CS}(6, null, \{\ldots, Spill(\overline{5})[Filled(4)], Dry(0, floor),$$
$$\underline{\neg Dry_c(6, floor)[Filled(4)]}, Filled(0), Drop(\overline{4}, ball)\}),$$
$$\mathbf{Proj}(6, null, \{\ldots, Dry(1:\infty), Filled(1:\infty)\})$$ [12]

The RMP rule is used in extending the context set. This allows Dudley to compute the extended effects of actions. It also allows him to deduce the future consequences of his planning as it interacts possibly with the actions of other agents or with events observed in the world. It allows for reasoning with the current projection by letting earlier events play out their consequences in an anticipated future before later events.

## 3.3 Context set extension and revision rule (CSR):

The CSR rule ensures that the context set is always kept updated to match the most current projection, and the state of the world in which the agent is situated. As explained before, formulas are annotated by the projections which are used to support them in future conjectures. In the event that the projections cease to hold as of "now", the formulas that are supported by them are dropped from the context set in the revision process. The revision is a kind of real-time truth maintenance. The CSR rule also plays the important role of resolving contradictions in a time situated manner.

[10] There is an implicit causality assumption here; earlier events are potential causes in axioms for changes but later events are useful only in explanations of past values, not responsible for changing the past values. Note that we say a *potential* point of change.

[11] This is an extended effect of the spill; we will elaborate later on the significance of this type of reasoning.

[12] Note that there is a contradiction between the **CS** and the **Proj**. It is resolved in the next step.

Following is a description of the rule used in deciding the contents of $\mathbf{CS_{i+1,p}}$ based on the contents of $\mathbf{CS_{i,p}}$ , $\mathbf{Proj_{i,p}}$ and $\mathbf{Ppl_{i,p}}$. In Part I we decide a set $\mathbf{Candid_{i,p}}$ selected from $\mathbf{CS_{i,p}}$ which are formulas to be considered as candidates for retention. Part II decides which members of $\mathbf{Candid_{i,p}}$ will make it to $\mathbf{CS_{i+1,p}}$.

*Part I (Select candidate formulas to inherit):*

1. If two formulas $\alpha$ and $\delta$ in $\mathbf{CS_{i,p}}$ du-contradict each other, then the following criteria are used in deciding which of them go to $\mathbf{Candid_{i,p}}$[13].
   (a) If $\alpha$ is a fact, while $\delta$ is a default (is annotated with a projection), select $\alpha$ and reject $\delta$ to go into $\mathbf{Candid_{i,p}}$.
   (b) If $\alpha$ and $\delta$ are both defaults, select neither[14] to go into $\mathbf{Candid_{i,p}}$.
2. Formulas that are not part of a contradiction go into $\mathbf{Candid_{i,p}}$.

*Part II (Choose among candidate formulas):*

1. A formula $\alpha$ from $\mathbf{Candid_{i,p}}$ which is a fact is inherited to $\mathbf{CS_{i+1,p}}$.
2. A formula $\alpha[\beta_1, \beta_2, \ldots, \beta_k]$ which is a default is inherited unless for some $1 \leq j \leq k$, $\beta_j \notin \mathbf{Proj_{i,p}}$. Also, if any of $\beta_1, \ldots, \beta_k$ now appear as facts in $\mathbf{CS_{i,p}}$, they are removed from the annotation.
3. Formulas corresponding to actions that are added to the plan in the previous step are added to the $\mathbf{CS_{i+1,p}}$[15].

We remark here, that there are two more rules which fire to add formulas to $\mathbf{CS_{i+1,p}}$. One is the OBS rule which add new observations that are made in step $i+1$ to $\mathbf{CS_{i+1,p}}$. The other is RMP, which was described before in detail.

**Example**
In step 5 in this example, Dudley concludes that there must have been a spill at step 5 based on the projection that the bucket was still filled at step 4. At the same time, however, he is told by a reliable observer that the bucket was in fact not filled at step 4 and adopts it as a fact.

The projection catches up at step 6 to no longer believe $Filled(4)$. As a result of CSR, $Spill(\overline{5})[Filled(4)]$ and $\neg Dry_c(6, floor)[Filled(4)]$ are no longer inherited to the context set at step 7. Note that the projection at step 7 already reflects a wet floor. This will also be subsequently revised in step 8 by an application of the TP rule, since $\neg Dry_c(6, floor)[Filled(4)]$ is no longer in $\mathbf{CS_{7,null}}$.

[13] Note that we do not encounter situations in which facts (direct or indirect descendents of observations alone) contradict each other. Observations with different time intervals involving the same predicate may well disagree, but these are not contradictory.

[14] Where both are defeasible beliefs, a working strategy is to not inherit either of them, and to continue the reasoning to see if one of them will reappear in the face of stronger evidence.

[15] Formulae in the context set are in fact, doubly annotated in the implementation, with the projections if any used in their derivation, and with the action(s) in the plan that are used in their derivation. Should the plan get revised to no longer require any of these actions, the corresponding formula is not inherited in the CS.

$$\mathbf{4}: \mathbf{CS}(4, null, \{\ldots, Dry(0, floor), Filled(0), \underline{Drop(\overline{4}, ball)}\}),$$
$$\mathbf{Proj}(4, null, \{\ldots, Dry(1:\infty, floor), Filled(1:\infty)\})$$

$$\mathbf{5}: \mathbf{CS}(5, null, \{\ldots, \underline{Spill(\overline{5})[Filled(4)]}, Dry(0, floor), Filled(0), \underline{\neg Filled(4)}, Drop(\overline{4}, ball)\}),$$
$$\mathbf{Proj}(5, null, \{\ldots, Dry(1:\infty, floor), Filled(1:\infty)\})$$

$$\mathbf{6}: \mathbf{CS}(6, null, \{\ldots, Spill(\overline{5})[Filled(4)], Dry(0, floor), Filled(0), \neg Filled(4),$$
$$\underline{\neg Dry_c(6, floor)[Filled(4)]}, Drop(\overline{4}, ball)\}),$$
$$\mathbf{Proj}(6, null, \{\ldots, Dry(1:\infty, floor), \underline{\neg Filled(5:\infty)}\})$$

$$\mathbf{7}: \mathbf{CS}(7, null, \{\ldots, Dry(0, floor), Filled(0), \neg Filled(4), Drop(\overline{4}, ball)\}),$$
$$\mathbf{Proj}(7, null, \{\ldots, \underline{Dry(1:5, floor)}, \underline{\neg Dry(7:\infty, floor)}, \neg Filled(5:\infty)\})$$

$$\mathbf{8}: \mathbf{CS}(8, null, \{\ldots, Dry(0, floor), Filled(0), \neg Filled(4), Drop(\overline{4}, ball)\}),$$
$$\mathbf{Proj}(8, null, \{\ldots, \underline{Dry(1:\infty, floor)}, \neg Filled(5:\infty)\})$$
$$\vdots$$

# 4 Time-situated Variations of the YSP

A time-situated reasoning mechanism was described in the previous section. Other aspects of the planner related to time estimates and feasibility analysis with respect to tight deadlines are described in [21]. We demonstrate here how this mechanism applies to several real-time variations of the Yale Shooting Problem [10] appropriate to active logics. The first is a *witness* scenario where Dudley is a witness to the scene of the crime. In it we show how Dudley draws the intuitive conclusion that Fred must be dead on observing a shoot action, and discard the unintuitive one where the gun mysteriously gets unloaded just before the shooting. We present two developments of the witness scenario which are of a *detective* nature where Dudley must offer a reasonable explanation about actions in the past, to fit his present observations. On seeing Fred alive at a later time, the same mechanism allows him to continue to perform belief revision to account for "why things went wrong"[19]. The last is a *killer* scenario where Dudley formulates a plan to kill Fred by a certain deadline and reasons that Fred is expected to be dead in the context of his plan to carry out a shoot action.

In the *classical* YSP problem, there is a certain ambiguity about the role of the reasoner. There the reasoning is itself timeless, presumably it takes place after all the events in question. Our treatment is significant in that Dudley the reasoner can reason *in* time about the events in progress and adjust his reasoning to suit new observations.

**Witness:** We suppose that the reasoner is an eyewitness on the scene of the crime: Dudley sees Fred and sees a loaded gun at time 0, but no action is observed. Then, following a wait period during which nothing happens, at time 4 Dudley sees the gun being fired at Fred, but cannot see what happens to Fred after that. Dudley is then draws the commonsense conclusion that Fred has been killed. Here we are mimicking Baker's simplified version of the YSP (no loading action[10] is required, and the *wait* action occurs between steps 0 and 4).

Below we sketch the steps in this reasoning.

**Axioms**(These are part of every context set):
$\neg Loaded(T) \vee \neg Shoot(\overline{T}) \vee \neg Alive(T+1)$;
$\neg Alive(T) \vee Alive(0:T)$

**0:** $\mathbf{CS}(0, null, \{\underline{Alive(0)_{obs}}, \underline{Loaded(0)_{obs}}\})$,
$\mathbf{Proj}(0, null, \{\})$

Dudley observes that a gun is loaded, and that Fred is alive. Dudley is a passive eyewitness. Hence the reasoning context is that of a *null* plan. There is no projection yet regarding either *Alive* or *Loaded*.

**1:** $\mathbf{CS}(1, null, \{Alive(0)_{obs}, Loaded(0)_{obs}\})$,
$\mathbf{Proj}(1, null, \{\underline{Alive(1:\infty)}, \underline{Loaded(1:\infty)}\})$

Rules yielding new conclusions: TP. There are no new observations.

**2:** $\mathbf{CS}(2, null, \{Alive(0)_{obs}, Loaded(0)_{obs}\})$,
$\mathbf{Proj}(2, null, \{Alive(1:\infty), Loaded(1:\infty)\})$

Wait period. No new actions or conclusions.

**3:** $\mathbf{CS}(3, null, \{Alive(0)_{obs}, Loaded(0)_{obs}\})$,
$\mathbf{Proj}(3, null, \{Alive(1:\infty), Loaded(1:\infty)\})$

Wait period.

**4:** $\mathbf{CS}(4, null, \{Alive(0)_{obs}, Loaded(0)_{obs}, \underline{Shoot(\overline{4})_{obs}}\})$,
$\mathbf{Proj}(4, null, \{Alive(1:\infty), Loaded(1:\infty)\})$

Rules yielding new conclusions: OBS. A shooting is observed by Dudley.

**5:** $\mathbf{CS}(5, null, \{Alive(0)_{obs}, \underline{\neg Alive_c(5)[Loaded(4)]}, Loaded(0)_{obs}, Shoot(\overline{4})_{obs}\})$,
$\mathbf{Proj}(5, null, \{Alive(1:\infty), Loaded(1:\infty)\})$

Rules yielding new conclusions: RMP. In the RMP application to the axiom $\neg Loaded(T) \vee \neg Shoot(\overline{T}) \vee \neg Alive(T+1)$ in clause form, $Shoot(\overline{4})$ is resolved with first since it is a fact. Next, $Loaded(4)$ is used in favor of $alive(5)$ in the resolution due to the projection being at an earlier time. It is allowed to play its effects before the projection at the next time is considered. The projection $Loaded(4)$ used in the inference is used to annotate the inference $\neg Alive_c(5)[Loaded(4)]$, and the result is noted as a possible point of inflection in the value of the predicate *Alive*.

**6:** $\mathbf{CS}(6, null, \{Alive(0)_{obs}, \neg Alive_c(5)[Loaded(4)], Loaded(0)_{obs}, Shoot(\overline{4})_{obs}\})$,
$\mathbf{Proj}(6, null, \{\underline{Alive(1:4)}, \underline{\neg Alive(6:\infty)}, Loaded(1:\infty)\})$

Rules yielding new conclusions: TP. Note that *Alive* is no longer projected to infinity but only until time 4 according to the new context set information. Also $\neg Alive$ is projected from time 6 onwards. The projection for *Loaded* remains unchanged.

⋮

The witness version of the YSP gives the intuitive answer: In the context of the *null* plan, Fred must have died at step 5 as a result of the shooting, provided of course, that the default regarding the gun staying loaded up until step 4 is indeed true. $\neg Alive(5)[Loaded(4)]$ is still defeasible, only as good as $Loaded(4)$ really, and is treated as a default.

**Detective:** To illustrate the real-time nature of our reasoning process, we now consider two different developments of the witness scenario following step 6. In one of them, at step 7, Fred is seen to be alive and walking about, in another, at step 7, the gun is examined, and found to be in fact unloaded. Suppose that the shooting itself does not unload the gun. What conclusions can Dudley make in each of these developments of the former scenario? In short, how can Dudley play detective in real-time?

### Development 1: But Fred is alive!

**7:** $\mathbf{CS}(7, null, \{\ldots, Loaded(0)_{obs}, Shoot(\overline{4})_{obs}, Alive(0)_{obs}, \neg Alive_c(5)[Loaded(4)], \underline{Alive(7)_{obs}}\}),$
$\mathbf{Proj}(7, null, \{Alive(1:4), \neg Alive(6:\infty), Loaded(1:\infty)\})$

Rules yielding new conclusions: OBS. Dudley sees that Fred is alive.

**8:** $\mathbf{CS}(8, null, \{\ldots, Loaded(0)_{obs}, Shoot(\overline{4})_{obs}$
$Alive(0)_{obs}, \underline{Alive(1:6)}, \neg Alive_c(5)[Loaded(4)], Alive(7)_{obs}\})$
$\mathbf{Proj}(8, null, \{\underline{Alive(1:4)}, \underline{Alive(8:\infty)}, Loaded(1:\infty)\})$

Rules yielding new conclusions: RMP, TP. $Alive(1:6)$ is derived by applying RMP to $\neg Alive(T) \vee Alive(0:T)$ in clause form. $Alive(7)$ being a fact, is used. The TP rule tries to make the best of all the *Alive* formulas in the context set to yield a projection. It will straighten out in the next step.

**9:** $\mathbf{CS}(9, null, \{\ldots, Loaded(0)_{obs}, \underline{\neg Loaded(4)}, Shoot(\overline{4})_{obs}, Alive(0)_{obs}, Alive(1:6), Alive(7)_{obs}\}),$
$\mathbf{Proj}(9, null, \{\underline{Alive(8:\infty)}, Loaded(1:\infty)\})$

Rules used to yield new conclusion: RMP, CSR, TP. The RMP rule derives $\neg Loaded(4)$ from $Alive(5)$, $Shoot(\overline{4})$ and $\neg Loaded(T) \vee \neg Shoot(\overline{T}) \vee \neg Alive(T+1)$. The CSR rule must deal with a contradiction in the context set. $Alive(5)$ is a fact, and $\neg Alive_c(5)[Loaded(4)]$ is a default. $Alive(5)$ makes it into the $\mathbf{Candid_{8,null}}$ set and is subsequently inherited, while the default is rejected. The TP rule also confronts the contradiction, but handles it with ease, since $Alive(0:7)$ are all facts. The projection is resumed quickly past the contradiction due to the presence of later facts.

**10:** $\mathbf{CS}(10, null, \{\ldots, Loaded(0)_{obs}, \neg Loaded(4), Shoot(\overline{4})_{obs}, Alive(0)_{obs}, Alive(1:6), Alive(7)_{obs}\}),$
$\mathbf{Proj}(10, null, \{\underline{Alive(8:\infty)}, \underline{\neg Loaded(5:\infty)}\})$

Rules yielding new conclusions: TP. The TP rule changes the projection to reflect the new knowledge that the gun was unloaded at step 4. Note, that no projections are made on whether the gun was loaded at steps 2 or 3.

⋮

### Development 2: Look at this gun, it is not loaded!

**7:** $\mathbf{CS}(7, null, \{Loaded(0)_{obs}, \underline{\neg Loaded(7)_{obs}} Shoot(\overline{4})_{obs},$
$Alive(0)_{obs}, \neg Alive_c(5)[Loaded(4)], \}),$
$\mathbf{Proj}(7, null, \{Alive(1:4), \neg Alive(6:\infty), Loaded(1:\infty)\})$

Rules yielding new conclusions: OBS. In this development of the eyewitness scenario, Dudley observes that the gun is unloaded at step 7. This reduces the feasibility of the *Loaded*(4) default, and Dudley's conclusions take a more skeptical turn. Note that there is already a disagreement between the formulas in $\mathbf{Proj_{7,null}}$ and $\mathbf{CS_{7,null}}$. It takes the agent one step to get the projection to match the changed context set.

**8:** $\mathbf{CS}(8, null, \{Loaded(0)_{obs}, \neg Loaded(7)_{obs}, Shoot(\overline{4})_{obs},$
$Alive(0)_{obs}, \neg Alive_c(5)[Loaded(4)], \}),$
$\mathbf{Proj}(8, null, \{Alive(1:4), \neg Alive(6:\infty), \underline{\neg Loaded(8:\infty)}\})$

Rules yielding new conclusions: TP. The projection is revised. Since 7 is not believed to be a point of change of the *Loaded* predicate (Dudley merely stumbles upon the unloaded gun, he does not see it being unloaded), no conjecture about *Loaded* is made over the 1 : 6 interval.

**9:** $\mathbf{CS}(9, null, \{Loaded(0)_{obs}, \neg Loaded(7)_{obs}, Shoot(\overline{4})_{obs}, Alive(0)_{obs}, \}),$
$\mathbf{Proj}(9, null, \{Alive(1:4), \neg Alive(6:\infty), \neg Loaded(8:\infty)\})$

Rules yielding new conclusions: CSR. $\neg Alive_c(5)[Loaded(4)]$ is not inherited since the projection in the annotation is not supported at the current step.

**10:** $\mathbf{CS}(10, null, \{Loaded(0)_{obs}, \neg Loaded(7)_{obs}, Shoot(\overline{4})_{obs}, Alive(0)_{obs}\}),$
$\mathbf{Proj}(10, null, \{\underline{Alive(1:\infty)}, \neg Loaded(8:\infty)\})$

Rules yielding new conclusions: TP. The Projection adjusts to the change in the *Alive* formulas in the context set. In the face of the observation that the gun was not loaded at step 7, Dudley reserves the conclusion regarding Fred's death from the shooting[16]. Should more evidence become available at a later time, this conclusion may be reinforced again.

$\vdots$

Both the above are explanation scenarios. The elegant recovery of the agent is to be attributed to the non-monotonic inference process that initiates changes to restore consistency. In the first development, when Fred is found to be alive past the time of the shooting, Dudley can successfully conclude that the gun must have been unloaded between the loading and the time of the shooting. Furthermore, *he can successfully change his on-going reasoning model to reflect these changes.* In the second development, when the gun is found to be unloaded at a later time, under the assumption that shooting does not unload the gun, Dudley chooses to be ambivalent about Fred's death. Even though he had imagined earlier that Fred would be dead, now he no longer has the same confidence in the projection, it must be changed, and the conclusion about Fred's death refrained from making, until more is known about the time of the unloading.

**Killer:** At time 0, Dudley sees that there is a loaded gun, and that Fred is alive. He is playing the killer, and has the goal $\neg Alive(5)$. He must formulate a plan to kill Fred. Our temporal projection mechanism must allow Dudley to support the intuitive extension that Fred will be dead in the context of Dudley's plan. This is crucial for Dudley to 'imagine' the unfolding of his plans. We show a few illustrative steps from this killer scenario:

[16]Since Dudley doesn't know when the gun was unloaded, he doesn't conclude $\neg Loaded(4)[Alive(5)]$ or $\neg Alive(5)[loaded(4)]$. $Loaded(0)$ is not projected due to the fourth item of the TP rule, since 7 is not believed to be a point of change of the *Loaded* predicate, as explained above. The conclusion $\neg Loaded(4)[Alive(5)]$ is prevented by the fourth item of the RMP rule, i.e., if a projection $\alpha$ with a later time than the time of $\beta$ is used in the RMP application, $\beta$ is not added to the context set.

**0:** $\mathbf{Goal}(kill, \neg Alive(5), 5)$,

$\mathbf{Proj}(0, null, \{\})$,

$\mathbf{CS}(0, null, \{\underline{Alive(0)_{obs}}, \underline{Loaded(0)_{obs}}\})$

Dudley's goal results in the formation of a partial plan. In this paper we do not describe the details of plan formation. The context set of the partial plan in the first step in its formation is the context set of the null plan. Also, the first projection is the projection in the null context.

**1:** $\mathbf{Goal}(kill, \neg Alive(5), 5)$,

$$\mathbf{Ppl}(1, kill, \left\{ \begin{bmatrix} Loaded(4) \\ Shoot(\overline{4}) \\ \neg Alive(5) \end{bmatrix}_1 \right\})$$

$\mathbf{CS}(1, null, \{Alive(0)_{obs}, Loaded(0)_{obs}\})$,

$\mathbf{Proj}(1, null, \{\underline{Alive(1:\infty)}, \underline{Loaded(1:\infty)}\})$,

$\mathbf{CS}(1, kill, \{\underline{Alive(0)_{obs}}, \underline{Loaded(0)_{obs}}\})$

$\mathbf{Proj}(1, kill, \{\underline{Alive(1:\infty)}, \underline{Loaded(1:\infty)}\})$,

Dudley formulates a plan to achieve the goal.

**2:** $\mathbf{Goal}(kill, \neg Alive(5), 5)$,

$$\mathbf{Ppl}(2, kill, \left\{ \begin{bmatrix} Loaded(4) \\ Shoot(\overline{4}) \\ \neg Alive(5) \end{bmatrix}_1 \right\})$$

$\mathbf{CS}(2, null, \{Alive(0)_{obs}, Loaded(0)_{obs}\})$,

$\mathbf{Proj}(2, null, \{Alive(1:\infty), Loaded(1:\infty)\})$,

$\mathbf{CS}(2, kill, \{Alive(0)_{obs}, \underline{Shoot(\overline{4})}, Loaded(0)_{obs}\})$,

$\mathbf{Proj}(2, kill, \{Alive(1:\infty), Loaded(1:\infty)\})$

$Shoot(\overline{4})$ is added to the context of the "kill" plan, since the action was added to the plan at the previous step.

**3:** $\mathbf{Goal}(kill, \neg Alive(5), 5)$

$$\mathbf{Ppl}(3, kill, \left\{ \begin{bmatrix} Loaded(4) \\ Shoot(\overline{4}) \\ \neg Alive(5) \end{bmatrix}_1 \right\})$$

$\mathbf{CS}(3, null, \{Alive(0)_{obs}, Loaded(0)_{obs}\})$,

$\mathbf{Proj}(3, null, \{Alive(1:\infty), Loaded(1:\infty)\})$

$\mathbf{CS}(3, kill, \{Alive(0)_{obs}, \underline{\neg Alive_c(5)[Loaded(4)]}, Shoot(\overline{4}), Loaded(0)_{obs}\})$,

$\mathbf{Proj}(3, kill, \{Alive(1:\infty), Loaded(1:\infty)\})$

In the context of his plan to kill, Dudley concludes that Fred will die. Within the context of his plans, his actions are treated as facts. It will take it another step to reach the same conclusion in the *null* plan.

**4:** $\mathbf{Goal}(kill, \neg Alive(5), 5)$, $\mathbf{Ppl}(4, kill, \underline{\{\}})$,

$\mathbf{CS}(4, null, \{Alive(0)_{obs}, Loaded(0)_{obs}, \underline{Shoot(\overline{4})_{obs}}\})$,

$\mathbf{Proj}(4, null, \{Alive(1:\infty), Loaded(1:\infty)\})$

$\mathbf{CS}(4, kill, \{Alive(0)_{obs}, Shoot(\overline{4})_{obs}, \neg Alive_c(5)[Loaded(4)], Loaded(0)_{obs}\})$,

$\mathbf{Proj}(4, kill, \{\underline{Alive(1:4)}, \underline{\neg Alive(6:\infty)}, Loaded(1:\infty)\})$

Dudley has acted upon his 'kill' plan, and the action triplet is removed from the partial plan, in accordance with the inference rules for planning and acting (not described here). Shoot action is observed, and appears in the context of both plans. In the context of his plan to kill, he concludes that Fred is dead from the shooting provided the projection that the gun stayed loaded is true.

⋮

# 5 Imagination and Two More Aspects of the Frame Problem

To imagine a state of the world in the future involves creating a mental image for which finding a real-time solution to three familiar aspects of the frame problem is essential. The temporal persistence rule (TP) tackles the projection frame problem, in deciding what fluents persist in the context of a given plan and for how long. The Context Set Revision rule (CSR), together with the restructured Modus Ponens rule (RMP) deals with the ramification problem, by allowing the agent to reason about the extended effects of its actions in the context of its plans. This allows Dudley to form a comprehensive mental image of the future and to extend it in time in accordance with his changing environment. The qualification problem must also be addressed if Dudley must anticipate the success of his actions in a plan. The treatment of various versions of the YSP illustrated Dudley's ability to handle the temporal persistence problem. In this section we examine how the active-logic planning mechanism offers solutions to the qualification and ramification problems.

## 5.1 The qualification problem

In a nutshell, the qualification problem is the impossibility of specifying all the (pre)-conditions required for an action. A famous example (due to McCarthy) is the 'banana in the tailpipe' problem. There are an infinite number of eventualities that can potentially interfere in the successful execution of an action toward its intended effect. It is impossible for the reasoner to account for all of these during the plan formulation or execution phases.

Apparently, in case of human commonsense reasoners, the knowledge representation for an action is determined by a complex learning mechanism, but once developed, the agent simply maintains a short list of preconditions that he/she deems necessary for the 'normal' execution of an action and takes them into account while formulating his/her plans. This has the tremendous advantage of keeping the planning process simple. But agents do not function in a static world. Hence the plans are not crisp. The agent is constantly observing and 'thinking' about his/her plan in the interval between formulation and execution. He/she has a great deal of world knowledge that makes it possible to recognize the presence of qualifications. Commonsense agents can incorporate these qualifications into the reasoning to alter their plans in suitable ways. The 'normal' conditions for an action are those that the agent is accustomed to accounting for while forming one's plan.

There are two aspects of the qualification problem. One falls into the category of learning. This is the problem of identifying whether a given condition is a qualification for a particular action. For example, a child learns through (repeated) trials that an object is impossible to lift with its fingers when it is hot. Thus $Hot(T, X)$ is a potential hindrance to the attempted action $Lift(T, object)$. Until the condition is not known to be a qualification (the axiom $Hot(T, object) \rightarrow Hinders(Lift(T, object))$ is not known), the agent attempts to lift the hot object and subsequently fails, unaware of the qualification. Thus agents need suitable learning capabilities to identify qualifications. The other aspect falls within the realm of planning. This is the problem of incorporating the *known qualifications* into the planning process *when considered necessary.* The human commonsense agent goes about its way, thinking only of the 'normal' conditions until some pointers to a potential qualification are seen in his/her knowledge base. Our real-time automated reasoner plans in a changing world. Observations can trigger further inferencing. Especially in deadline situations, these may be very crucial to making timely modifications to one's plans.

Consider the banana in the tailpipe. Suppose human agent Steve drives to work everyday. As he steps out of the house towards the car, Steve thinks of the only condition in his knowledge representation for the action of starting the engine – he just fumbles in his pocket to make sure he has the keys. If he just did not know that having a banana in the tailpipe can hinder the action of starting the car, he might think nothing of it even if he sees one sticking prominently in the tailpipe. This of course, will lead to subsequent failure to start the car, and maybe particularly expensive in a deadline situation. However, if he knows that the banana is a hindrance to the action of starting the car, and he notices it in the tailpipe, Steve, as a commonsense human agent immediately realizes that the plan of starting the car will not work. it needs modification. Possible modifications may include efforts to remove one or more of the responsible causes, such as getting rid of the banana, or may involve alternate planning to achieve the goal such as deciding to walk to work instead.

We would like our automated agent to have the capability to take care of a known qualification in its planning. The context set of the plan is continuously revised by the agent. As long as there is no formula in the context set indicating a hindrance to any of the actions in the plan, the agent concludes that it is ok to go ahead with the current action. This may lead to failure under some circumstances. However, in others, with the added real-time capacity to take notice of changes in one's environment, the agent may notice a qualification in time to modify the plan suitably.

We assume that Dudley has an inference rule that enables him to replan once a hindrance has been noticed. Here we only encapsulate it in the phrase *modify plan.* The details of the replanning have been omitted.

**Inference rule to take note of a hindrance:**

$$\frac{i : CS(i, p, \{\ldots, Hinders(A), \ldots\}), Ppl(i, p, \{\ldots, [C_A, A, R_A], \ldots\})}{i+1 : \mathit{modify\ plan\ p}}$$

**Axioms** (these are part of every context set):

$\neg Start(T : \overline{T+1}, engine) \vee Running(T+1, engine)$

$condition(Start(T : \overline{T+1}, engine), Have(T, key))$

$\neg In(T, banana, tailpipe) \vee \neg Start(T : \overline{T+1}, engine) \vee Hinders(Start(T : \overline{T+1}, engine))$

**0:** $\textbf{Goal}(car\_start, Running(6, engine), 6)$,
$\textbf{Proj}(0, null, \{\})$,
$\textbf{CS}(0, null, \{\underline{Have(0, key)_{obs}}\})$

**1:** $\textbf{Goal}(car\_start, Running(6, engine), 6)$,

$$\textbf{Ppl}\left(1, car\_start, \left\{ \begin{bmatrix} Have(5, key) \\ Start(5 : \overline{6}, engine) \\ Running(6, engine) \end{bmatrix}_1 \right\}\right),$$

$\textbf{CS}(car\_start, 1, \{Have(0, key)_{obs}\})$, $\textbf{Proj}(1, car\_start, \{\})$

**2:** **Goal**$(car\_start, Running(6, engine), 6)$,

$$\mathbf{Ppl}(2, car\_start, \left\{\begin{bmatrix} Have(5, key) \\ Start(5:\overline{6}, engine) \\ Running(6, engine)\end{bmatrix}_1\right\}),$$

$\mathbf{CS}(2, car\_start, \{Have(0, key)_{obs}, \underline{Start(5:\overline{6}, engine)}, \underline{In(2, banana, tailpipe)_{obs}}\})$,
$\mathbf{Proj}(2, car\_start, \{\underline{Have(1:\infty, key)}\})$

**3:** **Goal**$(car\_start, Running(6, engine), 6)$,
$\mathbf{CS}(3, car\_start, \{Have(0, key)_{obs}, Start(5:\overline{6}, engine),$
$\underline{Running_c(6, engine)}, In(2, banana, tailpipe)_{obs}\})$,
$\mathbf{Proj}(3, car\_start, \{Have(1:\infty, key), \underline{In(3:\infty, banana, tailpipe)}\})$

$$\mathbf{Ppl}(3, car\_start, \left\{\begin{bmatrix} Have(5, key) \\ Start(5:\overline{6}, engine) \\ Running(6, engine)\end{bmatrix}_1\right\}),$$

**4:** **Goal**$(car\_start, Running(6, engine), 6)$,
$\mathbf{CS}(4, car\_start, \{Have(0, key)_{obs}, Start(5:\overline{6}, engine), Running_c(6, engine\},$
$In(2, banana, tailpipe)_{obs}, \underline{Hinders(Start(5:\overline{6}, engine))[In(6, banana, tailpipe)]},$
$\mathbf{Proj}(4, car\_start, \{Have(1:\infty, key), In(3:\infty, banana, tailpipe),$
$Running(6:\infty, engine)\})$

$$\mathbf{Ppl}(4, car\_start, \left\{\begin{bmatrix} Have(5, key) \\ Start(5:\overline{6}, engine) \\ Running(6, engine)\end{bmatrix}_1\right\}),$$

At step 4, the context set of plan *car_start* contains $Hinders(Start(5:\overline{6}, engine))[In(6, banana, tailpipe)]$. The agent notices this in step 5, and moves on to modify the plan, either by removing the $Start(5:\overline{6}, engine)$ action from the plan and finding an alternate one, or by removing the cause of the hindrance. We do not include details of the replanning flowing this step, since the focus of this work is the frame and temporal issues.

A relevance mechanism brings into short term focus only those conditions that may be relevant to the processing of an action at the current time. This is achieved by integrating a model of a short-term memory and an algorithm guided by heuristic rules for bringing those formulas which are relevant to the agents current line of reasoning. For example, if the agent is not currently aware of a banana in the tailpipe, the particular axiom regarding the hindrance is not in focus, hence not in the current context set. Thus, the above formalism, combined with a relevance mechanism offers a fuller treatment of the qualification problem. Details of the space constrained agent are in [22].

## 5.2 The ramification problem

This is the difficulty in explicitly recording all the consequences of an action. An action often produces, in addition to its intended effect, a variety of ramifications (or side-effects) that are often dependent on the environment in which the action is performed, upon the domain (integrity) constraints existing at the time, etc. STRIPS like systems could not handle this problem at all. Effects of actions were canned into add and delete lists. We would like the automated reasoner to have the flexibility to reason about extended effects that are within the realm of its inference capacity and permitted under his deadline constraints. We illustrate this with the help of an extension of the example scenario mentioned before in section 3. To recapitulate: Dudley is standing next to a bucket, at location *bucket_loc*, with a ball in his hand. The bucket is filled to the brim with oil. Dudley may be the household robot assigned with normal clean up goals, and routinely drops objects into buckets (trash cans). In his 'normal' sphere of activity he seldom encounters buckets filled with oil. Dudley has the compound goal $In(5, ball, bucket) \wedge at(10, Dudley, nextroom)$, i.e., to drop the ball in the bucket and then go to the next room. His knowledge representation of the $Drop(\overline{T}, Object)$ action is determined by his everyday experience. It has the condition $At(T, Dudley, bucket\_loc) \wedge Have(Dudley, Object)$ and has the result $In(T+1, bucket, Object)$. However, when Dudley drops the ball in the bucket, the oil spills all over the floor, and hinders Dudley's next action of walking to the next room, since he slips on the oil and falls.

Can Dudley reason about this obvious extended ramification of his drop action? He knows that the bucket is full. He has the axiom that dropping an object will cause the oil in the bucket to spill. The spilled oil will cause the floor to get wet, and will hinder his walk. As in the above scenario, ramifications are of particular interest when the ramification of one action (Drop) becomes a qualification for another (Walk)[17].

Our representation of the contexts of the plans, and the continued reasoning within the context allows Dudley to take care of obvious ramifications of his actions.

**Axioms:**
$\neg Drop(T : \overline{T+1}, ball, bucket) \vee In(T+1, ball, bucket)$
$condition(Drop(T : \overline{T+1}, ball, bucket), At(T, dudley, bucket\_loc) \wedge have(T, ball, dudley))$
$\neg Filled(T, bucket, oil) \vee \neg Drop(T : \overline{T+1}, ball) \vee Spill(\overline{T+1}, oil)$
$\neg Spill(T : \overline{T+1}, oil) \vee \neg Dry(T+1, floor)$
$\neg Walk(\overline{T1 : T2}, dudley, L1 : L2) \vee At(15, dudley, L2)$ ( (T2 -T1)v = L2 - L1, where v is Dudley's speed.)
$condition(Walk(\overline{T1 : T2}, dudley, L1 : L2), At(15, dudley, L1))$
$Dry(T, floor) \vee \neg Walk(S : F, dudley) \vee Hinders(Walk(S : F, dudley)), S \leq T \leq F$

**5:** $\mathbf{Goal}(p, In(15, ball, bucket) \wedge At(20, dudley, nextroom, 20)$,

$$\mathbf{Ppl}(5, p, \left\{ \begin{array}{c} \left[ \begin{array}{c} At(14, dudley, bucket\_loc) \wedge Have(14, dudley, ball) \\ Drop(14 : \overline{15}, ball) \\ In(15, ball, bucket) \end{array} \right]_1 \\ \left[ \begin{array}{c} At(15, dudley, bucket\_loc) \\ Walk(15 : \overline{18}, dudley, bucket\_loc : nextroom) \\ At(18, dudley, nextroom) \end{array} \right]_2 \end{array} \right\})$$

$\mathbf{CS}(5, p, \{Filled(0, bucket)_{obs}, Have(0, ball, dudley)_{obs}, At(0, dudley, bucket\_loc)_{obs}, Dry(0, floor)_{obs}, Drop(14 : \overline{15}, ball), In_c(15, ball, bucket), Walk(15 : \overline{18}, dudley, bucket\_loc : nextroom), At_c(18, dudley, nextroom)\})$
$\mathbf{Proj}(5, p, \{Filled(1 : \infty, bucket), Have(1 : \infty, ball, dudley), At(1 : 17, dudley, bucket\_loc), At(18 : \infty, dudley, nextroom), Dry(1 : \infty, floor), In(16 : \infty, ball, bucket)\})$

The above step in reasoning shows a plan to achieve the goal, and shows its context set at step 5, soon after plan formulation. In steps 6, 7, and 8 to follow, by applications of RMP, $Spill(\overline{15}, oil)$, $\neg Dry(16, floor)$ and $Hinders(Walk(15 : \overline{18}, dudley, bucket\_loc : nextroom))$ will be members of the $Context\_List$. Once 'Hinders' enters the context of

[17]With suitable knowledge about the domain, our mechanism will also allow us to deal with actions in the self-defeating category [9]. However these problems involve the characterization of the applicable physical laws, and tend to become quite complex in any framework.

plan p, Dudley must take the required replanning steps to remedy the situation. One possible remedy could be to wait until the floor dries, but the agent must evaluate this plan against the possibility of overshooting the deadline for his goal. Thus in deadline situations, it is not sufficient for the situated agent to detect the ramifications, it also needs to have the ability to react to them in a timely fashion. This example illustrates a real-time treatment of the ramification problem.

# 6 Comparison with Related Work

We have presented a framework that handles several real-time versions of the YSP, addresses several aspects of the frame problem, and performs fully deadline-coupled planning and reasoning, while accounting for the passage of time in the face of an approaching deadline. As such, it is difficult to compare it with isolated brittle solutions that each handle particular temporal aspects of one of the above mentioned problems. Moreover, the inference engine was not designed to solve the YSP or to tackle particular aspects of the frame problem. Instead, we consider these problems as benchmarks against which to test our flexible and yet robust real-time reasoning mechanism.

What distinguishes this work is the capacity of an 'active' agent to perform the reasoning in a *changing* environment, accounting for new observations, and incorporating them into the current beliefs. The agent dynamically forms a model to fit his/her evolving knowledge base by using a combination of procedural and declarative methods. There is no meta-reasoner to sneak out and perform the difficult tasks, neither are undecidable methods used to compute all plausible models. The agent effectively computes 'a' plausible model, and has the capacity to revise it as time goes on, resolving contradictions as and when they arise in the process. The reasoning proceeds in time, and itself consumes time. The mechanism functions for the agent whether he/she is the one performing the actions, or observing them.

Numerous solutions have been proposed to the YSP; [7, 8, 11, 12, 15, 14, 20, 23, 25, 2] to mention a few. Broadly, the solutions may be grouped into three categories:

1. chronological minimization [25, 12],
2. causal minimization [11, 15], and
3. model approach [19, 1].

The chronological minimization approaches were criticized by Kautz [12], in that, by always granting preference to later changes over earlier ones, the agent may draw artificial conclusions about the time of an action that caused a change. This is highlighted in the 'stolen car' problem, where the agent is forced to conclude that the car was stolen only at the last possible moment before it was found missing.

Our solution to the YSP has commonalities with certain of these approaches, notably those of Shoham [25], Lifschitz [14], and Kautz [12]. In our approach as well as theirs, defaults are applied forward in time, so that earlier events play out their consequences for later ones. However, these approaches can not solve either the detective scenario or the 'stolen car' problems. In short, they specialize in temporal projection problems, but can not solve explanation problems.

We prefer a different term for our approach: *parallel projection*. That is, the entire known state of the world at one moment is used to determine the (expected) state at the next moment. Since active logics are built around the idea of specifying what is known (proven) *so far*, all predicates (e.g., *Alive* and *Loaded*) can be simultaneously reconsidered at each new time step. Persistence of *Alive* from one moment to another depends on everything that is (known to be) true at the earlier moment, because in fact that is how things are really affected in the world. This is also the central idea behind the RMP rule. Our mechanism does not run into a problem with the 'stolen car'. Suppose it is known that $\neg Missing(0, car)$ and then, at time 10, Dudley observes $Missing(10, car)$, the TP rule does not project $\neg Missing$ up until time 9, unless $Missing_c(10)$ is known. We do not postpone changes until the last possible moment unless a potential cause for the change is known. We have the versatility to handle both projection and explanation problems as shown by the detective scenarios.

The causal minimization approaches [11, 15] introduce a special "Causes" predicate which they then minimize using circumscription. A drawback is that the axioms can not be written in an unrestricted logic. Changes that have known causes are preferred over unjustifiable changes. However, this means that when no knowledge about a cause exists, changes can not be reasoned with correctly. Lifschitz [14] introduces "miracles" to get around this difficulty to solve the 'stolen car' problem. Uncertainty is tackled to some extent in [11] through the notion of 'potential causes' when the agent knows about the occurrence of an action but is uncertain about its success. We do not yet have a mechanism to handle this type of uncertainty (we assume that all actions in the agent's plan succeed unless observed otherwise).

Among the other original approaches, Morgenstern and Stein [19] provide an elegant solution to the YSP, and also provide a solution to Development 1 in the 'detective' scenario, in which Fred is observed to be alive after the shooting, so that what went wrong must be figured. However, in their solution, there is no time adjoint to the reasoning; a meta reasoner concludes (supposedly sometime after all the observations) that the gun must have been unloaded before the shooting. They select models using the criterion of 'fewer unmotivated actions'. However, there is no acting agent who can actually compute these models and check which one is preferred. It would be interesting to devise an inference rule for the active logic whereby an agent applies their criterion to break ties and resolve contradictions in real-time. We would like to explore this approach in future work. Since there is no role attached to the reasoner in [19], there is no treatment of the 'killer' case.

Baker [2] provides a solution to the YSP using situation calculus and circumscription, with an 'Ab' predicate in which the result is varied. It is necessary to add "existence" axioms for situations, but this is done elegantly in this framework. Baker's approach can solve most versions of the YSP (though not in real-time). It is not clear who does the reasoning, the (undecidable) circumscription provides the likely models. Since the agent's role is not obvious, it is not possible to integrate it within a planning framework, hence it can not be easily extended to the "killer" scenario. It can not solve the 'stolen car' problem, which we can address. We can handle all the other problems mentioned in [2], although our detective scenario is closer to that in [19] and different from the murder mystery described. Our approach can offer a treatment for all the ramification problems mentioned in [2].

Amsterdam's [1] is the only other work that attempts to discuss the issue of who the reasoner is in a given scenario, which we have addressed at length. It highlights the advantage which a reasoner has when he/she is at the site of the action, namely, that he/she can observe an action whenever it happens. This allows him to formulate the closure property – "if an action is not mentioned then it did not happen". However, we remark that once again, there is a meta reasoner doing the inference. The theory's greatest limitation (and this is said in the work itself) is the rigidity of the above mentioned closure principle. There are scenarios where actions can be derived from propositions and hence do not have to be explicitly specified. Dudley does not suffer from this problem.

# 7 Conclusion

We have described the real-time reasoning process of a commonsense agent in a tight deadline situation, with emphasis on the nonmonotonic temporal reasoning aspects. We have demonstrated how the active-logic planning mechanism successfully handles some interesting real-time variants of the Yale shooting problem. Two other aspects of the frame problem, namely, the qualification and ramification problems are also dealt with in the same framework. Importantly, these techniques are employed by the agent himself, without the aid of any meta reasoning processes whose time is not accounted for. Having once drawn a conclusion (which may be a default), added evidences resulting in apparent contradictions do not swamp Dudley with an infinite confusion of wffs. Unlike traditional logic based approaches, he can sort his way through the initial contradiction in time.

We have shown how Dudley can successfully handle temporal explanation problems with the same degree of ease as temporal projection. As an agent functioning in the real world, Dudley can (simultaneously) perform the role of an observer or a planner. Our solutions to each of these illustrate the agent's ability to form *contexts* within which to reason, to *project* in each context thus formed by applying default inferences, and to *revise* and extend its conclusions within each context by applying time-sensitive inference rules, and most importantly, to account for *all* the time spent in the process. The active logic planner exhibits qualities of an agent capable of real-time imagination in a formal, and primarily deliberative setting.

# References


[1] J. Amsterdam. Temporal reasoning and narrative conventions. In *Proc. of KR-91*, pages 15–21, 1991.

[2] Andrew B. Baker. A simple solution to the Yale Shooting Problem. In *Proceedings, KR89*, pages 11–20, 1989.

[3] P. Cohen and H. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.

[4] T. Dean and D. McDermott. Temporal data base management. *Artificial Intelligence*, 32(1):1–55, 1987.

[5] J. Elgot-Drapkin. *Step-logic: Reasoning Situated in Time.* PhD thesis, Department of Computer Science, University of Maryland, College Park, Maryland, 1988.

[6] J. Elgot-Drapkin and D. Perlis. Reasoning situated in time I: Basic concepts. *Journal of Experimental and Theoretical Artificial Intelligence*, 2(1):75–98, 1990.

[7] Michael Gelfond. Autoepistemic logic and formalization of common-sense reasoning. In *Proceedings of the second international workshop on nonmonotonic reasoning*, 1988. Munich 88.

[8] Michael P. Georgeff. Many agents are better than one. In F. M. Brown, editor, *Proceedings of the 1987 Workshop on The Frame Problem*, pages 59–75x. Morgan Kaufmann, 1987. Lawrence, Kansas.

[9] M. L. Ginsberg and D. E. Smith. Reasoning about action I: A possible worlds approach. In F. M. Brown, editor, *Proceedings of the 1987 Workshop on The Frame Problem*, pages 233–258, Lawrence, KS, 1987. Morgan Kaufmann.

[10] S. Hanks and D. McDermott. Nonmonotonic logic and temporal projection. *Artificial Intelligence*, 33:379–412, 1987.

[11] Brian A. Haugh. Simple causal minimizations for temporal persistence and projection. In *Proceedings of the sixth national conference on artificial intelligence*, pages 218–223, 1987.

[12] H. Kautz. The logic of persistence. In *Proceedings, AAAI-86*, pages 401–405, 1986.

[13] S. Kraus, M. Nirkhe, and D. Perlis. Deadline-coupled real-time planning. In *Proceedings of 1990 DARPA workshop on Innovative Approaches to Planning, Scheduling and Control*, pages 100–108, San Diego, CA, 1990.

[14] V. Lifschitz. Pointwise circumscription. In M. Ginsberg, editor, *Readings in Nonmonotonic Reasoning*, pages 179–193. Morgan Kaufmann, 1987.

[15] Vladimir Lifschitz. Formal theories of action. In *The Frame Problem in Artificial Intelligence*, pages 35–57, Los Altos, CA, 1987. Morgan-Kaufmann.

[16] J. McCarthy and P. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 4, pages 463–502. Edinburgh University Press, 1969.

[17] D. McDermott. Planning and acting. *Cognitive Science*, 2:71–109, 1978.

[18] D. McDermott. Nonmonotonic logic and temporal projection. *Artificial Intelligence*, 33:379–412, 1987.

[19] L. Morgenstern and L. A. Stein. Why things go wrong: A formal theory of causal reasoning. In *Proceeding, AAAI-88*, pages 518–523, 1988.

[20] P. H. Morris. The anomalous extension problem in default reasoning. *Artificial Intelligence*, 35:383–399, 1988.

[21] M. Nirkhe, S. Kraus, and D. Perlis. Fully deadline-coupled planning: One step at a time. In *Proceedings of the sixth international symposium on methodologies for intelligent systems*, Charlotte, NC, 1991.

[22] M. Nirkhe, S. Kraus, and D. Perlis. Situated reasoning within tight deadlines and realistic space and computation bounds. In *Proceedings of the Second Symposium On Logical Formalizations Of Commonsense Reasoning*, Austin, TX, 1993.

[23] Judea Pearl. On logic and probability. *Computational Intelligence*, 4:99–103, 1988.

[24] D. Perlis, J. Elgot-Drapkin, and M. Miller. Stop the world! – I want to think! Invited paper, *International J. of Intelligent Systems*, special issue on Temporal Reasoning, K. Ford and F. Anger (eds.), vol. 6, 1991, pp. 443–456.

[25] Yoav Shoham. Chronological ignorance: Experiments in nonmonotonic temporal reasoning. *Artificial Intelligence*, 36:279–331, 1988.

[26] R. Wilensky. *Planning and understanding.* Addison Wesley, Reading, Mass, 1983.