Appeared in Renee Elio, editor, *Proceedings of the Tenth Biennial Conference of the Canadian Society for Computational Studies of Intelligence,* pp. 271-277, Banff, Canada, May, 1994.

An Argument for Indexical Representations in Temporal Reasoning*

Yves Lespérance and Hector J. Levesque†
Department of Computer Science
University of Toronto
Toronto, ON, M5S 1A4 Canada
{lesperan,hector}@ai.toronto.edu

Abstract

This paper discusses the need for indexicals in a representation language. It has been claimed that the cost of updating a knowledge base containing indexicals would be prohibitive and thus that a robot should use its internal clock to eliminate indexicals from its representations. We criticize this view and give an example of a commonplace temporal reasoning/planning problem that can only be solved in a representation formalism that includes both indexical and absolute terms and supports reasoning using both. We show that the example can be formalized within our theory of knowledge and action. We argue that rather than trying to find restricted settings where indexical knowledge can be reduced to objective knowledge, one should investigate when and how planning and temporal knowledge base update can be performed efficiently in the presence of indexicals.

*This research received financial support from the Institute for Robotics and Intelligent Systems (Canada), the Information Technology Research Center (Ontario, Canada), and the Natural Science and Engineering Research Council (Canada).
†Fellow of the Canadian Institute for Advanced Research

1 Introduction

To someone accustomed to the objective point of view of science or mathematics, indexicality (context-sensitivity) may appear as little more than an artifact of natural language. One may thus claim that while using indexical descriptions (e.g., *now, three hours ago, one meter in front of me*) is often convenient, in practice, indexical knowledge can always be understood objectively. One reason for wishing that this claim were true has to do with the fact that the semantic content of indexical representations depends on the context, so if the context changes you may have to adjust the representations to keep the semantic content unchanged. For instance, if an agent's knowledge base describes some facts as holding 'now', then at the next time step, it should describe these facts as holding 'one time step before now'.[1] Haas [1991] points out that the cost of adjusting a knowledge base that contains indexical time references for the passage of time would be high, if implemented in the obvious way. He proposes that a robot use its internal clock to eliminate all occurrences of 'now' in its representations.

This proposal and the general strategy of trying to eliminate indexical representations are misguided.[2] First, humans do not have internal clocks that they can use in the way a robot can, and they do not always know what time it is. A system that is interacting with humans will need to model this (e.g. to remind a user that his meeting is just starting). Even if we limit ourselves to simple robotics contexts, it seems unlikely that the internal clocks of robots could always be guaranteed to be accurate. In such cases, Haas's scheme leads to indexical information being misrepresented. Moreover, Haas's robot cannot even represent the fact that his internal clock is incorrect; it could not monitor for this and plan to get its clock reset when appropriate. Also, for very simple (e.g. insect-like) robots, the cost of fitting them with internal clocks and setting them may be too high. Finally, as Haas recognizes, it is not at all clear that the cost of updating a temporal knowledge base that contains indexicals need be prohibitive; for example, if

[1] Subramanian and Woodfill [1989] prove that such a transformation is truth-preserving within their indexical situation calculus framework.

[2] Philosophers such as Perry [1979] have long argued that indexical knowledge cannot be eliminated. But the examples they cite to support their position can appear farfetched. Here we try to make the point that this does apply to the practice of building intelligent systems.

all occurrences of 'now' are replaced by a new constant and the fact that this new constant is equal to 'now' is added, then only this single assertion need be updated as time passes.

In the next section, we will give an example of a commonplace temporal reasoning/planning problem that can only be solved in a representation formalism that *includes both indexical and non-indexical concepts and supports reasoning using both.* The example involves an agent that does not initially know what time it is; he must keep track of time in relative terms (using a timer), but later convert this indexical knowledge into absolute knowledge for communication to another agent. We will show in detail in Section 4 that this example can be formalized within the theory of knowledge and action that we proposed in [Lespérance and Levesque, 1994; Lespérance, 1991]. We will return to our discussion of whether indexical knowledge can be eliminated in the final section.

2 The Example

The example goes as follows. Imagine that you arrive at home and are greeted by the following note:

> "The turkey is ready to go into the oven (at 325°F). I will be home in time to take it out, but leave me a message before you go telling me at what time you put it in."

Unfortunately, the only timing devices you have are the one-hour timer on the stove, and a radio station that announces the time at least every 30 minutes. You also want to put the turkey to roast as early as possible and be done as quickly as possible.

It is not too hard to see what needs to be done. A reasonable plan is to put the turkey in the oven, set the timer to one hour, then listen to the radio until the time is announced while keeping track of the roasting time with the timer, and finally calculate the time the turkey started roasting, and leave a message to that effect.[3]

Obviously, a single example by itself is not much of an argument for anything. However, the example illustrates a simple but extremely common

[3]It would be more efficient to listen to the radio *as* one is putting the turkey in, but we do not want to deal with true concurrency here.

situation where two very different notions of time need to be dealt with: indexical or relative time, as determined by the timer ("the turkey went in 20 minutes ago"), and absolute or objective time, as calculated from the radio announcement ("the turkey went in at 12:45pm"). The key point is that either notion of time *by itself* is inadequate to model the situation. To see this, observe that the problem cannot be solved without the radio, and yet all it provides is information like "It is now 1:05pm." To make sense of this essential piece of information, we need to be able to relate the two separate conceptions of time. In purely indexical or purely absolute terms, the information is meaningless.

3 Overview of the Formalism

Before presenting a formalization of the example, we briefly review a formal theory of indexical knowledge, action and ability (see [Lespérance and Levesque, 1994; Lespérance, 1991] for a more detailed presentation.) The goal is to be able to express attributions of indexical knowledge, for example, that Rob knows that he himself was holding a cup five minutes ago.[4] In such cases, what is known is a "proposition" that is relative. It may be relative to the knower, or to the time of the knowing, or perhaps to other aspects of the context. To handle this, our language includes two special terms: **self**, which denotes the current agent, and **now**, which denotes the current time; these terms are called *primitive indexicals*. Non-logical (domain-dependent) symbols may also depend on the current agent and time for their interpretation, for example, $\exists x \textsc{Holding}(x)$ may express the fact that the current agent is currently holding something — we say that such symbols are *non-primitive indexicals*. Our semantics handles this by interpreting terms and formulas with respect to *indices*, which consist of a possible-world (modeling the objective circumstances), and an agent and a time (modeling the context).

The language used is a many-sorted first-order modal language with equality called *LIKA* (Language of Indexical Knowledge and Action). It includes terms of four different sorts: terms for ordinary individuals (as usual), temporal terms, agent terms, and action terms. For each of these sorts, there

[4]As well, we want to be able to distinguish this from having objective knowledge, such as Rob's knowing that Rob was holding a cup at some specified time, say, 4:37pm.

are both variables and function symbols (i.e., functions whose values are of the proper sort); as usual, constants are taken to be 0-ary function symbols.

The atomic formulas include predications using predicate symbols and terms, written $R(\theta_1, \ldots \theta_n)$, which are used to assert that $\theta_1, \ldots \theta_n$ stand in static relation R at the current time for the current agent. We also have equality expressions $(\theta_1 = \theta_2)$, between terms of the same sort, as well as expressions of temporal precedence $(\theta_1^t < \theta_2^t)$. We assume that time is linearly ordered. Finally, $\mathbf{Does}(\theta^d, \theta^t)$ is used to assert that the current agent does action θ^d starting from the current time and ending at time θ^t.

Non-atomic formulas may be composed using the standard boolean connectives and quantifiers as well as a set of modal operators. $\mathbf{At}(\theta^t, \varphi)$ means that φ holds at time θ^t, that is, when θ^t is taken to be the current time. $\mathbf{By}(\theta^a, \varphi)$ means that φ holds when θ^a is taken to be the current agent. $\Box\varphi$ means that φ is historically necessary at the current time, that is, that φ now holds in all possible courses of events that are identical to the current one up to the current time. We also introduce a dual to $\Box$: $\Diamond\varphi \stackrel{\text{def}}{=} \neg\Box\neg\varphi$.

$\mathbf{Know}(\varphi)$ is used to represent the fact that the current agent knows at the current time that φ. If φ contains indexical elements, $\mathbf{Know}(\varphi)$ should be taken as attributing indexical knowledge, that is, knowledge the agent has about himself and the current time. For example, $\mathbf{Know}(\textsc{Holding}(x))$ could mean that the agent knows that *he himself* is *currently* holding the object denoted by x. The semantics for **Know** is a simple generalization of the standard possible-world scheme. The knowledge accessibility relation K is taken to hold over indices rather than plain possible worlds. Informally, $\langle\langle \mathsf{w}, \mathsf{a}, \mathsf{t}\rangle, \langle \mathsf{w}', \mathsf{a}', \mathsf{t}'\rangle\rangle \in \mathsf{K}$ if and only if as far as agent a at time t in world w knows, it may be the case that w' is the way the world actually is *and* he is a' *and* the current time is t'. Thus, we allow an agent to be uncertain not only about what world he is in, but also about who he is and what time it is. We assume that the knowledge accessibility relation K is reflexive and transitive, meaning that **Know** obeys the principles of modal logic S4. We also assume that agents have perfect memory and always know what actions they have done.

For the example to follow, it is convenient to make two general assumptions beyond those embodied in the logic. First, we assume that the domain of times is (or is isomorphic to) the integers, that is, we assume that the facts about integer arithmetic that we need are valid and that constants and function symbols representing arithmetic operations are rigid. Secondly, we assume that all actions are "solid", in the sense that any overlapping instances of an action (type) must be the same instance (have the same endpoints) [Shoham, 1987]. This ensures that we can refer to things like "the starting time of the action I have just done".

To talk more easily about a wider class of actions, it is useful to extend the use of **Does** to a new syntactic category, that of *action expressions*. These include action terms as above, which represent simple actions, **noOp**, which represents the empty action and takes no time, $(\delta_1;\delta_2)$, which represents the sequential composition of the actions δ_1 and δ_2, and $\mathbf{if}(\varphi,\delta_1,\delta_2)$, which represents the action that consists in doing action δ_1 if the condition φ holds, and in doing action δ_2 otherwise. Formulas of the form $\mathbf{Does}(\delta,\theta^t)$ where δ is an action expression, can be thought of as abbreviations that reduce to formulas where **Does** ranges only over the simple action terms, in the obvious way. We also define a bounded form of "while loop" as an action expression as follows:

$$\mathbf{while}_{\mathsf{k}}(\varphi,\delta) \stackrel{\text{def}}{=} \begin{cases} \mathbf{noOp} & \text{if } \mathsf{k}=0 \\ \mathbf{if}(\varphi,(\delta;\mathbf{while}_{\mathsf{k-1}}(\varphi,\delta)),\mathbf{noOp}) & \text{if } \mathsf{k}>0, \mathsf{k}\in\mathbb{N} \end{cases}$$

Let us also define some dynamic-logic-style operators that will be used in our formalization of ability. $\mathbf{AfterNec}(\delta,\varphi)$, which is intended to mean "φ must hold after δ", is defined inductively as follows:

- $\mathbf{AfterNec}(\theta^d,\varphi) \stackrel{\text{def}}{=} \Box\forall v^t(\mathbf{Does}(\theta^d,v^t) \supset \mathbf{At}(v^t,\varphi))$, where v^t is a temporal variable that does not occur free in φ
- $\mathbf{AfterNec}(\mathbf{noOp},\varphi) \stackrel{\text{def}}{=} \varphi$
- $\mathbf{AfterNec}((\delta_1;\delta_2),\varphi) \stackrel{\text{def}}{=} \mathbf{AfterNec}(\delta_1,\mathbf{AfterNec}(\delta_2,\varphi))$
- $\mathbf{AfterNec}(\mathbf{if}(\varphi_c,\delta_1,\delta_2),\varphi) \stackrel{\text{def}}{=}$
  $(\varphi_c \supset \mathbf{AfterNec}(\delta_1,\varphi)) \wedge (\neg\varphi_c \supset \mathbf{AfterNec}(\delta_2,\varphi))$

Also, let $\mathbf{PhyPoss}(\delta) \stackrel{\text{def}}{=} \neg\mathbf{AfterNec}(\delta,\mathbf{False})$. $\mathbf{PhyPoss}(\delta)$ is intended to mean that it is "physically possible" for **self** to do action δ next (even though he may not be able to do it because he does not know what primitive actions δ stands for). **True** (**False**) stands for some tautology (contradiction).

Our formalization of ability, based on that of Moore [1980], says that the agent is able to achieve the goal φ by doing action δ, formally $\mathbf{Can}(\delta,\varphi)$, if and only if he can do action δ and knows that after doing δ, the goal φ must hold:

$$\mathbf{Can}(\delta, \varphi) \stackrel{\text{def}}{=} \mathbf{CanDo}(\delta) \wedge \mathbf{Know}(\mathbf{AfterNec}(\delta, \varphi))$$

$\mathbf{CanDo}(\delta)$ is defined inductively as follows:[5]

- $\mathbf{CanDo}(\theta^d) \stackrel{\text{def}}{=} \exists v^d\ \mathbf{Know}(v^d = \theta^d) \wedge \mathbf{Know}(\mathbf{PhyPoss}(\theta^d))$, where action variable v^d does not occur free in θ^d
- $\mathbf{CanDo}(\mathbf{noOp}) \stackrel{\text{def}}{=} \mathbf{True}$
- $\mathbf{CanDo}(\delta_1; \delta_2) \stackrel{\text{def}}{=} \mathbf{Can}(\delta_1, \mathbf{CanDo}(\delta_2))$
- $\mathbf{CanDo}(\mathbf{if}(\varphi, \delta_1, \delta_2)) \stackrel{\text{def}}{=}$
  $(\mathbf{Know}(\varphi) \wedge \mathbf{CanDo}(\delta_1)) \vee (\mathbf{Know}(\neg\varphi_c) \wedge \mathbf{CanDo}(\delta_2))$

Note that we eliminate Moore's requirement that the agent know who he is; instead, we require indexical knowledge (see [Lespérance and Levesque, 1994] for a discussion of why this is better). Also, as we will see in the next section, the fact that our account of ability is based on a more expressive temporal logic allows it to deal with actions whose prerequisites or effects involve knowledge of absolute times and knowing what time it is.

4 Formalizing the Example

Let us formalize the example and prove that the agent is able to achieve his goal by executing the proposed plan given some reasonable assumptions about his initial knowledge. Our formalization will be rather simplistic, but could easily be made more accurate and general. We define the agent's plan as follows:

$$\textsc{getDinnerGoing} \stackrel{\text{def}}{=} \textsc{startRoasting}; \textsc{setTimer}(1\textsc{h}); \textsc{listenUntilTimeAnnounced}; \textsc{lookAtTimer}; \textsc{leaveMsg}$$

[5]This way of defining **Can** is preferable to the one in [Lespérance, 1991; Lespérance and Levesque, 1990] as it separates the knowledge prerequisites involving the goal from the rest. The definitions of **AfterNec** and **PhyPoss** given here are also changed; they now behave exactly as their dynamic logic [Goldblatt, 1987] counterparts do.

We use abbreviations such as 1H (1 hour) and 30MIN (30 minutes) for readability; it is assumed that such abbreviations stand for the corresponding number of seconds. The complex action LISTENUNTILTIMEANNOUNCED is defined below. Let ANNOUNCEDTIME mean that the time has been announced on the radio during the last step of LISTEN:[6]

$$\begin{aligned}\textsc{AnnouncedTime} \stackrel{\text{def}}{=} & \exists t_s \exists t_i(\mathbf{DoesFromTo}(\textsc{listen}, t_s, \mathbf{now}) \wedge \\ & t_s \leq t_i \leq \mathbf{now} \wedge \mathbf{At}(t_i, \textsc{AnnouncingTime}))\end{aligned}$$

LISTENUNTILTIMEANNOUNCED stands for repeatedly doing LISTEN until the time is announced (at least once and at most some very large number of times vln):

$$\begin{aligned}&\textsc{listenUntilTimeAnnounced} \stackrel{\text{def}}{=} \\ &\quad \textsc{listen}; \mathbf{while}_{\mathsf{vln}}(\neg\textsc{AnnouncedTime}, \textsc{listen})\end{aligned}$$

Let us now formalize the actions involved in the plan. First note that we must specify an appropriate limit on the duration of the actions, otherwise they might take so much time as to prevent the goal from being achieved. But note also that we cannot have these actions take a fixed length of time that is known to the agent, for otherwise, he could use them to measure time and dispense with the timer. What we will do is specify that basic actions take a indeterminate amount of time that must be between given bounds. So we specify the effects of the STARTROASTING action as follows:

Assumption 1 (Effects of STARTROASTING)

$$\begin{aligned}\models \forall t_s \forall t_e(&\mathbf{DoesFromTo}(\textsc{startRoasting}, t_s, t_e) \supset \\ & t_e = (t_s + 1\textsc{min}) \pm 30\textsc{s} \wedge \mathbf{At}(t_e, \textsc{Roasting}) \wedge \\ & \forall t(t_s < t < t_e \supset \mathbf{At}(t, \neg\textsc{Roasting})))\end{aligned}$$

This says that setting the turkey to roast takes one minute plus or minus 30 seconds (i.e., between half a minute and a minute and a half), that the turkey is roasting afterwards, and that it is not roasting while the action is being done.

For the action of setting the timer, we have the following:

[6] $\mathbf{DoesFromTo}(\delta, \theta^t_s, \theta^t_e)$ means that the agent does action δ from time θ^t_s to time θ^t_e; formally: $\mathbf{DoesFromTo}(\delta, \theta^t_s, \theta^t_e) \stackrel{\text{def}}{=} \exists v^t_e(v^t_e = \theta^t_e \wedge \mathbf{At}(\theta^t_s, \mathbf{Does}(\delta, v^t_e)))$, provided that v^t_e does not occur anywhere in θ^t_s, θ^t_e, or δ.

Assumption 2 (Effects of SETTIMER)

$$\models \forall t_s \forall t_e \forall n(\mathbf{DoesFromTo}(\text{SETTIMER}(n), t_s, t_e) \supset \\ t_e = (t_s + 15\text{S}) \pm 5\text{S} \wedge \mathbf{At}(t_e, \text{TIMERVAL} = n))$$

This says that doing $\text{SETTIMER}(n)$ takes 15 seconds plus or minus 5 seconds, and that afterwards, the timer is set to show that the time left is n seconds.

For the action of listening to the radio (LISTEN), we assume that it takes 10 seconds plus or minus 4 seconds, and that afterwards, the agent either knows that the time has not been announced during the action, or knows that it has been announced and knows what time it is within a margin of error of 7 seconds:

Assumption 3 (Effects of LISTEN)

$$\models \forall t_s \forall t_e(\mathbf{DoesFromTo}(\text{LISTEN}, t_s, t_e) \supset t_e = (t_s + 10\text{S}) \pm 4\text{S} \wedge \\ \mathbf{At}(t_e, \mathbf{Know}(\neg\text{ANNOUNCEDTIME}) \vee \\ (\mathbf{Know}(\text{ANNOUNCEDTIME}) \wedge \exists t \mathbf{Know}(\mathbf{now} = t \pm 7\text{S}))))$$

It is assumed that the time is announced on the radio at least every half hour.

For the action of looking at the timer, we assume that it takes 5 seconds plus or minus two seconds and that afterwards, the agent knows what duration is left on the timer:

Assumption 4 (Effects of LOOKATTIMER)

$$\models \forall t_s \forall t_e(\mathbf{DoesFromTo}(\text{LOOKATTIMER}, t_s, t_e) \supset \\ t_e = (t_s + 5\text{S}) \pm 2\text{S} \wedge \mathbf{At}(t_e, \exists n \mathbf{Know}(n = \text{TIMERVAL})))$$

Finally, we need to specify the effects of leaving a message. Let $\text{TSR}(\theta^t)$ stand for the claim that the turkey has been roasting since time θ^t and was not roasting prior to that:

$$\text{TSR}(\theta^t) \stackrel{\text{def}}{=} \forall v^t(\theta^t \leq v^t \leq \mathbf{now} \supset \mathbf{At}(v^t, \text{ROASTING})) \\ \wedge \mathbf{At}(\theta^t - 1\text{S}, \neg\text{ROASTING}), \text{ where } v^t \text{ is not free in } \theta^t$$

We will assume that LEAVEMSG, that is, leaving a message about what time the turkey started roasting, takes one minutes plus or minus 30 seconds, and that afterwards, there must be a message on the table stating that the turkey started roasting at some time t_m that is within ϵ seconds of the time at which the turkey actually started roasting:

Assumption 5 (Effects of LEAVEMSG)

$$\models \forall t_s \forall t_e(\mathbf{DoesFromTo}(\textsc{leaveMsg}, t_s, t_e) \supset t_e = (t_s + 1\textsc{min}) \pm 30\textsc{s} \\ \wedge\, \mathbf{At}(t_e, \exists t_r \exists t_m(\textsc{MsgOnTbl}(t_m) \wedge \mathrm{TSR}(t_r) \wedge t_m = t_r \pm \epsilon)))$$

For this example, the error bound ϵ can be made as tight as 14 seconds, but nothing depends crucially on this.

Now, let us specify the physical prerequisites of the actions. We assume that it is physically possible for the agent to do STARTROASTING whenever the turkey is not roasting:

Assumption 6 $\models \neg\textsc{Roasting} \supset \mathbf{PhyPoss}(\textsc{startRoasting})$

We also assume that it is physically possible for the agent to set the stove timer to any duration between 0 and 1 hour (the formal statement is similar to the one above). As well, it is assumed that LISTEN, LOOKATTIMER, and LEAVEMSG are always physically possible.

We must also specify the conditions under which agents know how to perform the basic actions. We assume that one always knows how to do STARTROASTING:

Assumption 7 $\models \exists d\mathbf{Know}(d = \textsc{startRoasting})$

Similarly, we assume that agents always know how to do LOOKATTIMER, LISTEN, and SETTIMER(n) (for any n). Finally, we assume that the agent must know how to do LEAVEMSG if he knows when the turkey started roasting within a margin of error of ϵ:

Assumption 8 (LEAVEMSG is known)

$$\models \exists t_m \mathbf{Know}(\exists t_r(\mathrm{TSR}(t_r) \wedge t_r = t_m \pm \epsilon)) \supset \exists d\mathbf{Know}(d = \textsc{leaveMsg}))$$

We also have various frame assumptions that specify what remains unchanged as actions are done. First, we assume that actions other than STARTROASTING have no effects on whether or not the turkey is roasting.[7] Secondly, we assume that for any action other than setting the timer, the time shown by the timer must accurately reflects the passage of time during the action:

[7]This frame assumption (and the subsequent ones) should not really be taken to hold for all actions. But given the limited domain under consideration, this causes no harm. Such assumptions would probably be best specified as default statements.

Assumption 9 (Frame assumption about the value of the timer)

$$\models \forall t_s \forall t_e \forall d \forall n \forall t_i (\mathbf{DoesFromTo}(d, t_s, t_e) \wedge \forall m (d \neq \textsc{setTimer}(m)) \wedge \\ \quad \mathbf{At}(t_s, n = \textsc{timerVal}) \wedge t_s < t_i \leq t_e \\ \quad \supset \mathbf{At}(t_i, \textsc{timerVal} = \textsc{max}(0, n - (t_i - t_s))))$$

Finally, we have unique name assumptions for all the actions introduced.

Given this formalization, our framework now allows us to prove the following proposition, which says that if the agent knows that the turkey is not yet roasting, then by doing the action $\textsc{getDinnerGoing}$, he is able to achieve the goal that there be a message on the table telling when the turkey started roasting within ϵ seconds of accuracy and that the time after the action be less than 32 minutes and 12 seconds after the turkey started roasting:

Proposition 1

$$\models \mathbf{Know}(\neg \textsc{Roasting}) \supset \\ \quad \mathbf{Can}(\textsc{getDinnerGoing}, \exists t_r \exists t_m (\textsc{MsgOnTbl}(t_m) \wedge \mathrm{TSR}(t_r) \wedge \\ \qquad t_r = t_m \pm \epsilon \wedge \mathbf{now} - t_r \leq 32\textsc{min}12\textsc{s}))$$

A sketch of the proof is provided in appendix.

5 Discussion

Let us return to our discussion of the claim that indexical knowledge can be reduced to objective knowledge. In our semantics for knowledge, indexical terms and formulas are treated as relative to an agent and a time; for example, knowing that something is **here** amounts to knowing that something is at one's position at the current time. Given this, it is clear that if one knows who one is and knows what time it is (we are taking this to require knowing a standard name), then anything that one knows in an indexical way is also known in an objective way. But is it reasonable to assume that an agent always knows who he is and what time it is?

As argued in section 1, the temporal part of this question must clearly be answered negatively.[8] Humans do not always know what time it is and

[8]We also think that it should not be assumed that agents always know who they are; see [Lespérance and Levesque, 1994; Lespérance, 1991; Grove and Halpern, 1991] for arguments.

computers need to model this. And even robots sometimes need to get their internal clocks reset. Work on reactive agent architectures supplies other reasons for wanting a formalism that can represent indexical knowledge. As pointed out by Agre and Chapman [1987], the world can change in unexpected ways and reasoning about change can be very costly; in some cases it is better to rely on perception to get fresh information at every time step rather than try to update a representation of the world; in such cases, the problem of updating indexical representations does not arise. And as Rosenschein and Kaelbling [1986] have shown, it is legitimate to ascribe knowledge to agents even when they have no explicit representation of this knowledge. In such cases, one needs a formalism that distinguishes between indexical and objective knowledge just to accurately model the agent's thinking. The output of the agent's perception module says nothing about time, and even if the agent has a correct internal clock, he may have no need to time-stamp his knowledge. We want a formalism that makes the distinctions required to model this.

Thus, rather than trying to find restricted settings where indexical knowledge can be reduced to objective knowledge, we think it would be more productive to investigate *when and how planning and temporal knowledge base update can be performed efficiently in the presence of indexicals.* Once one allows for agents not knowing what time it is, then examples like the one formalized here easily come to mind, examples that require both indexical and absolute terms for their representation and the ability to relate them in reasoning. A formalism along the lines of ours is required for handling such cases. Grove and Halpern's logic of knowledge [Grove and Halpern, 1991] handles some aspects of indexicality, but does not deal with time; so it cannot handle the kinds of situations discussed here. Subramanian and Woodfill's version of the situation calculus [Subramanian and Woodfill, 1989] handles aspects of indexicality, but not knowledge; thus, it cannot account for knowledge acquisition actions.

Let us conclude by discussing various areas in which our framework could be extended or improved. It may be possible to develop more convenient constructs for specifying of the temporal constraints associated with actions. We are currently reformulating our framework into an extended version of the situation calculus, to incorporate a solution to the frame problem described in [Scherl and Levesque, 1993]. We are also developing a more general account of the notions of "ability to achieve a goal" and "knowing how to execute a plan"

[Levesque *et al.*, 1994]. Other important issues are how default information could be specified, and identifying restrictions on domain theories and queries that guarantee tractability or decidability.

Acknowledgements

We would like to thank Andy Haas for his comments on the ideas advanced in this paper.

A Outline of the Proof of Proposition 1

The following lemmas are the main steps in proving proposition 1. One proves the proposition by "chaining" these lemmas using the following properties of **Can**:

$$\text{If } \models \varphi_i \supset \mathbf{Can}(\delta_2, \varphi_e), \text{ then } \models \mathbf{Can}(\delta_1, \varphi_i) \supset \mathbf{Can}((\delta_1; \delta_2), \varphi_e)$$

$$\models \mathbf{Can}(\delta, \varphi) \supset \mathbf{Can}(\delta, \mathbf{Know}(\varphi))$$

The first lemma shows that given his initial knowledge, the agent is able to set the turkey to roast:

Lemma 1

$\models \mathbf{Know}(\neg\textsc{Roasting}) \supset \mathbf{Can}(\textsc{startRoasting}, \mathrm{TSR}(\mathbf{now}))$

The proof uses assumptions 1, 6, and 7.

The second lemma shows that once he has set the turkey to roast, the agent is able to start measuring the roasting time by setting the timer to one hour. Let us first define MRT, meaning that the agent is measuring the roasting time:

$$\mathrm{MRT} \stackrel{\text{def}}{=} \exists t(\mathrm{TSR}(t) \wedge t = (\mathbf{now} - (1\textsc{h} - \textsc{timerVal}) - 15\textsc{s}) \pm 5\textsc{s})$$

We can then establish that:

Lemma 2

$\models \mathbf{Know}(\mathrm{TSR}(\mathbf{now})) \supset \mathbf{Can}(\textsc{setTimer}(1\textsc{h}), \mathrm{MRT} \wedge \textsc{timerVal} = 1\textsc{h})$

The proof uses assumption 2, the assumptions that $\textsc{setTimer}$ is always known and physically possible, the frame assumption for $\textsc{Roasting}$, and the unique name assumption for actions.

Then, we show that once he gets in that state, by doing the iterative action $\textsc{listenUntilTimeAnnounced}$ the agent can find out what time it is within 7 seconds of accuracy, with the timer still measuring the roasting time:

Lemma 3

$$\begin{aligned}
&\models \mathbf{Know}(\mathrm{MRT} \wedge \textsc{timerVal} = 1\textsc{h}) \supset \\
&\quad \mathbf{Can}(\textsc{listenUntilTimeAnnounced}, \\
&\qquad \mathrm{MRT} \wedge 30\textsc{min} - 14\textsc{s} < \textsc{timerVal} \wedge \exists t \mathbf{Know}(\mathbf{now} = t \pm 7\textsc{s}))
\end{aligned}$$

To prove this, we need the following two sublemmas. Let us define ATSTS, meaning that the time has been announced on the radio since the timer was set:

$$\begin{aligned}
&\mathrm{ATSTS} \stackrel{\mathrm{def}}{=} \\
&\quad \exists t(\mathbf{now} - (1\textsc{h} - \textsc{timerVal}) \leq t \leq \mathbf{now} \wedge \mathbf{At}(t, \textsc{AnnouncingTime}))
\end{aligned}$$

The first sublemma states that if the agent knows that he is measuring the roasting time with the timer and that the time has not been announced on the radio since he set the timer, then by doing $\textsc{listen}$, he is able to either find out that the time has been announced during the action and know what it is (within 7 seconds of accuracy), or know that it has not been announced during the action and since he set the timer, while continuing to measure the roasting time with the timer:

Lemma 4

$$\begin{aligned}
&\models \forall m(\mathbf{Know}(\mathrm{MRT} \wedge 30\textsc{min} < \textsc{timerVal} \leq m \wedge \neg\mathrm{ATSTS}) \supset \\
&\quad \mathbf{Can}(\textsc{listen}, \mathrm{MRT} \wedge \\
&\qquad ([\mathbf{Know}(\textsc{AnnouncedTime}) \wedge 30\textsc{min} - 14\textsc{s} < \textsc{timerVal} \wedge \\
&\qquad\ \ \exists t \mathbf{Know}(\mathbf{now} = t \pm 7\textsc{s})] \vee \\
&\qquad\ \ [\mathbf{Know}(\neg\textsc{AnnouncedTime}) \wedge 30\textsc{min} < \textsc{timerVal} \leq m - 6\textsc{s} \wedge \\
&\qquad\ \ \neg\mathrm{ATSTS}])))
\end{aligned}$$

This is proven using assumptions 3 and 9, the assumptions that $\textsc{listen}$ is always possible and known, the frame assumption for $\textsc{Roasting}$, the assumption that the time is announced at least every thirty minutes, and the unique name assumption. From the above lemma, we can then prove the following by induction over the bound on the number of iterations n:

Lemma 5

For all $\mathsf{n} \in \mathbb{N}$,

$$\begin{array}{l}\models \mathbf{Know}(\mathrm{MRT}) \wedge \\ \quad ([\mathbf{Know}(\textsc{AnnouncedTime} \wedge 30\textsc{min} - 14\textsc{s} < \textsc{timerVal}) \\ \qquad \wedge \exists t \mathbf{Know}(\mathbf{now} = t \pm 7\textsc{s})] \\ \qquad \vee \mathbf{Know}(\neg\textsc{AnnouncedTime} \wedge 30\textsc{min} < \textsc{timerVal} \leq 30\textsc{min} + \mathsf{n}6\textsc{s} \wedge \\ \qquad\qquad\quad \neg\mathrm{ATSTS})) \supset \\ \quad \mathbf{Can}(\mathbf{while}_{\mathsf{n}}(\neg\textsc{AnnouncedTime}, \textsc{listen}), \\ \qquad\quad \mathrm{MRT} \wedge 30\textsc{min} - 14\textsc{s} < \textsc{timerVal} \wedge \exists t \mathbf{Know}(\mathbf{now} = t \pm 7\textsc{s}))\end{array}$$

This says that if the agent either knows that the time has been announced during the previous LISTEN step and knows what it is (within 7 seconds), or knows that it has not been announced during the action and since he set the timer, while measuring the roasting time, then by repeatedly doing LISTEN until the time is announced, he is able to find out what time it is (within 7 seconds) while continuing to measure the roasting time. Lemma 3 is then proven by "chaining" the two results above.

Returning to the proof of the proposition, we then show that once he has found out the time, the agent can find out at what time the turkey started roasting by looking at the timer:

Lemma 6

$$\begin{array}{l}\models \mathbf{Know}(\mathrm{MRT} \wedge 30\textsc{min} - 14\textsc{s} < \textsc{timerVal}) \wedge \exists t \mathbf{Know}(\mathbf{now} = t \pm 7\textsc{s}) \supset \\ \quad \mathbf{Can}(\textsc{lookAtTimer}, \\ \qquad\quad \exists t_m \mathbf{Know}(\exists t_r(\mathrm{TSR}(t_r) \wedge t_r = t_m \pm 14\textsc{s} \wedge \mathbf{now} - t_r \leq 30\textsc{min}42\textsc{s})))\end{array}$$

This is shown using assumptions 4 and 9, the assumptions that LOOKATTIMER is always possible and always known, the frame assumption for ROASTING, and the unique name assumption.

Finally, we show that once he has found out when the turkey started roasting, by doing LEAVEMSG, the agent can ensure that there is a message on the table telling when the turkey started roasting (within 14 seconds of accuracy):

Lemma 7

$$\models \exists t_m \mathbf{Know}(\exists t_r(\mathrm{TSR}(t_r) \wedge t_r = t_m \pm 14\textsc{s} \wedge \mathbf{now} - t_r \leq 30\textsc{min}42\textsc{s})) \supset$$
$$\mathbf{Can}(\textsc{leaveMsg}, \exists t_r \exists t_m(\mathrm{TSR}(t_r) \wedge \textsc{MsgOnTbl}(t_m) \wedge t_r = t_m \pm 14\textsc{s} \wedge \mathbf{now} - t_r \leq 32\textsc{min}12\textsc{s}))$$

The proof uses assumptions 5 and 8, the assumption that LEAVEMSG is always physically possible, the frame assumption for ROASTING, and the unique name assumption.

References

[Agre and Chapman, 1987] Philip E. Agre and David Chapman. Pengi: An implementation of a theory of activity. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 268–272, Seattle, WA, July 1987. American Association for Artificial Intelligence, Morgan Kaufmann Publishing.

[Goldblatt, 1987] Robert Goldblatt. *Logics of Time and Computation*. CSLI Lecture Notes No. 7. Center for the Study of Language and Information, Stanford University, Stanford, CA, 1987.

[Grove and Halpern, 1991] Adam J. Grove and Joseph Y. Halpern. Naming and identity in a multi-agent epistemic logic. In James Allen, Richard Fikes, and Erik Sandewall, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*, pages 301–312, Cambridge, MA, 1991. Morgan Kaufmann Publishing.

[Haas, 1991] Andrew R. Haas. Indexical expressions and planning. Unpublished manuscript, Department of Computer Science, State University of New York, Albany, NY, 1991.

[Lespérance and Levesque, 1990] Yves Lespérance and Hector J. Levesque. Indexical knowledge in robot plans. In *Proceedings of the Eight National Conference on Artificial Intelligence*, pages 868–874, Boston, August 1990. American Association for Artificial Intelligence, AAAI Press/MIT Press.

[Lespérance and Levesque, 1994] Yves Lespérance and Hector J. Levesque. Indexical knowledge and robot action — a logical account. To appear in *Artificial Intelligence*, 1994.

[Lespérance, 1991] Yves Lespérance. *A Formal Theory of Indexical Knowledge and Action*. PhD thesis, Department of Computer Science, University of Toronto, Toronto, ON, January 1991. Also published as technical report CSRI-248.

[Levesque *et al.*, 1994] Hector J. Levesque, Yves Lespérance, Fangzhen Lin, and Richard B. Scherl. Knowledge, action, and ability in the situation calculus. In preparation, 1994.

[Moore, 1980] Robert C. Moore. Reasoning about knowledge and action. Technical Report 191, AI Center, SRI International, Menlo Park, CA, October 1980.

[Perry, 1979] John Perry. The problem of the essential indexical. *Noûs*, 13:3–21, 1979.

[Rosenschein and Kaelbling, 1986] Stanley J. Rosenschein and Leslie P. Kaelbling. The synthesis of digital machines with provable epistemic properties. In Joseph Y. Halpern, editor, *Theoretical Aspects of Reasoning about Knowledge: Proceedings of the 1986 Conference*, pages 83–98, Monterey, CA, 1986. Morgan Kaufmann Publishing.

[Scherl and Levesque, 1993] Richard B. Scherl and Hector J. Levesque. The frame problem and knowledge-producing actions. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 689–695, Washington, DC, July 1993. AAAI Press/The MIT Press.

[Shoham, 1987] Yoav Shoham. Temporal logics in AI: Semantical and ontological considerations. *Artificial Intelligence*, 33(1):89–104, 1987.

[Subramanian and Woodfill, 1989] Devika Subramanian and John Woodfill. Making the situation calculus indexical. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, pages 467–474, Toronto, ON, May 1989. Morgan Kaufmann Publishing.