A Knowledge-Theoretic Approach to Distributed Problem Solving

Michael Wooldridge

Department of Electronic Engineering, Queen Mary & Westfield College
University of London, London E1 4NS, United Kingdom

M.J.Wooldridge@qmw.ac.uk

Abstract. Traditional approaches to distributed problem solving have treated the problem as one of distributed search. In this paper, we propose an alternative, logic-based view of distributed problem solving, whereby agents cooperatively solve problems by exchanging information in order to derive the solution to a problem using logical deduction. In particular, we give a *knowledge theoretic* model of distributed problem solving, and show how various problem solving strategies can be represented within this scheme.

1 Introduction

Distributed problem solving is perhaps the paradigm example of activity in multi-agent systems [1]. It occurs when a group of logically decentralised agents cooperate to solve problems that are typically beyond the capabilities of any individual agent. Historically, distributed problem solving has been viewed and modelled as a kind of *distributed search*, whereby a collection of agents collaboratively traverse the search space of a problem in order to find a solution [10]. This model obviously mirrors the long-studied and well-understood view of problem solving as search from mainstream Artificial Intelligence (AI) [8].

In short, the purpose of this paper is to put forward an alternative, *logic-based* view of distributed problem solving [6]. In this view, distributed problem solving is treated as a *multi-agent deduction* problem. This viewpoint, while comparatively novel in multi-agent systems research, nevertheless echoes and builds upon the long and highly successful tradition of problem solving through theorem proving from mainstream AI [9].

The basic idea of the approach is both simple and intuitive. A problem to be solved is phrased as a question of logical consequence: does conclusion ψ follow from premises $\varphi_1, \ldots, \varphi_n$? In our model, the premises are distributed among a collection of *agents*. Each agent is equipped with some deductive capability and the ability to communicate. Problem solving proceeds by agents applying their deductive capability to the part of the problem they have been allocated, and sharing results with other agents by broadcasting them. The information that is shared in this way can then be used by recipients to derive further conclusions, and so on. Eventually, we hope, an agent will have sufficient information to derive the conclusion. In the traditional (centralised) view of theorem proving, the key question to be answered is what rule to apply next (and hence which lemma to prove next). In the multi-agent deduction view, they key question becomes *which message to send next*.

The particular emphasis of this paper is on a *knowledge theoretic* account of multi-agent problem solving [4]. Thus we begin in section 2 by defining a *temporal epistemic logic* that allows us to model both the information carried by agents (i.e., their part of the problem), and how this information evolves over time, as problem solving proceeds. The notion of a multi-agent system is defined in section 3, and in particular, this section shows how the temporal epistemic logic developed in section 2 can be used to represent the history of such a system. Section 4 introduces the notion of a problem, and defines what it means for a multi-agent system to solve a problem. Some basic results relating to problems and multi-agent systems are established in this section, and some practical multi-agent problem solving strategies are discussed. In particular, we show how a form of deduction closely related to classical resolution can be realised using the framework presented in this paper. Finally, section 5 presents some conclusions and discusses related research.

2 Logical Preliminaries

We begin by assuming a set $Ag = \{1, \ldots, n\}$ of *agents*, or more precisely, *agent identifiers*. We use i to stand for members of Ag. Next, we assume a finite vocabulary $\Phi = \{p, q, r, \ldots\}$ of *primitive propositions*. These are the atomic components of the languages we will use to express problems.

A *state* is defined to be a (possibly empty) subset of Φ. The idea is that a state explicitly identifies the propositions that are true in it. States will thus do service as propositional valuations, of the kind that are used in normal modal logics. They will allow us to do without such valuations in our framework. For example, if $s = \{p, q\}$, then we know that the only primitive propositions true in s are p and q. If $s' = \emptyset$, then every primitive proposition in s' is false. This approach is, of course, strictly less powerful than that of using valuation functions, since it implies that any two states are equal if agree on the valuation of primitive propositions (i.e., they contain the same elements). However, this is not a problem for our work. Let $S = \wp(\Phi)$ be the set of all states. We use s (with annotations: $s', s_1, \ldots$) to stand for members of S.

In order to express the properties of states, we introduce a classical propositional logic $\mathcal{L}_0$. This logic contains the classical connectives "$\wedge$" (and), "$\vee$" (or), "$\neg$" (not), "$\Rightarrow$" (implies), and "$\Leftrightarrow$" (if, and only if), as well as logical constants for truth ("**true**") and falsity ("**false**"). We define syntax and semantics for disjunction and negation, and assume the remaining connectives and constants are introduced as abbreviations in the standard way. Formally, the syntax of $\mathcal{L}_0$ is defined

ECAI 98. *13th European Conference on Artificial Intelligence*
Edited by Henri Prade
Published in 1998 by John Wiley & Sons, Ltd.

by the following grammar:

$$\langle \mathcal{L}_0\text{-}fmla \rangle \quad ::= \quad \text{any element of } \Phi \mid \mathbf{true} \mid \neg\langle \mathcal{L}_0\text{-}fmla \rangle \mid \langle \mathcal{L}_0\text{-}fmla \rangle \vee \langle \mathcal{L}_0\text{-}fmla \rangle$$

Let $wff(\mathcal{L}_0)$ be the set of (well-formed) formulae of $\mathcal{L}_0$. The semantics of $\mathcal{L}_0$ are defined via the $\mathcal{L}_0$ satisfaction relation "$\models_{\mathcal{L}_0}$", which holds between states and members of $wff(\mathcal{L}_0)$. The rules defining this relation are as follows:

$$\begin{array}{lll} s & \models_{\mathcal{L}_0} & p \quad \text{iff } p \in s \quad (\text{where } p \in \Phi) \\ s & \models_{\mathcal{L}_0} & \mathbf{true} \\ s & \models_{\mathcal{L}_0} & \neg\varphi \quad \text{iff not } s \models_{\mathcal{L}_0} \varphi \\ s & \models_{\mathcal{L}_0} & \varphi \vee \psi \quad \text{iff } s \models_{\mathcal{L}_0} \varphi \text{ or } s \models_{\mathcal{L}_0} \psi \end{array}$$

Next, we introduce *knowledge sets*, which in our formalism will play the role usually taken by accessibility relations in knowledge theory [4]. The idea, as in knowledge theory, is to characterise the information carried by an agent — its knowledge — as a set of states. Each state represents one way the world could be, given what the agent knows. However, rather than explicitly introducing a relation over states to characterise an agent's knowledge, we will instead simply represent it as a set of states. Although this technique is in principle less expressive that the traditional accessibility relation, it will not affect our formalism or our results. We let $KS = \wp(S)$ be the set of all knowledge sets, and use κ (with annotations: $\kappa', \kappa_1, \ldots$) to stand for members of KS. If $\varphi \in wff(\mathcal{L}_0)$, then we write κ_φ for the knowledge set that contains just those states that satisfy φ, i.e., $\kappa_\varphi = \{s \mid s \models_{\mathcal{L}_0} \varphi\}$.

Our definition of *knowledge* is essentially identical to that of knowledge theory: an agent $i \in Ag$ with knowledge set κ_i knows φ if φ is satisfied by all states in κ_i. For the purposes of this paper, we will only be concerned with knowledge that is expressed in $\mathcal{L}_0$: we will not be concerned with *nested* knowledge (i.e., knowledge about knowledge). This will be considered odd by readers familiar with normal modal (S5) epistemic logic [4], but nested knowledge it is not required for expressing our problems. We define a binary metalanguage predicate $knows \subseteq wff(\mathcal{L}_0) \times KS$ to capture our definition of knowledge:

$$knows(\varphi, \kappa) \text{ iff } \forall s \in \kappa, \text{ we have } s \models_{\mathcal{L}_0} \varphi.$$

To express the properties of knowledge sets, we introduce a multiagent epistemic logic, $\mathcal{L}_1$, which contains an indexed set of unary modal connectives K_i, one for each agent $i \in Ag$. A formula $K_i\varphi$ is to be read "agent i knows φ". In addition, $\mathcal{L}_1$ contains a *distributed knowledge* modality, D. A formula $D\varphi$ is to be read "there is distributed knowledge of φ". The intuitive semantics of this operator are that if $D\varphi$ then φ could be deduced by "pooling" the knowledge of all other agents. See, e.g, [4] for a discussion of distributed knowledge. Syntactically, $\mathcal{L}_1$ is defined by the following grammar:

$$\langle \mathcal{L}_1\text{-}fmla \rangle \quad ::= \quad K_i\langle \mathcal{L}_0\text{-}fmla \rangle \mid D\langle \mathcal{L}_0\text{-}fmla \rangle \mid \neg\langle \mathcal{L}_1\text{-}fmla \rangle \mid \langle \mathcal{L}_1\text{-}fmla \rangle \vee \langle \mathcal{L}_1\text{-}fmla \rangle$$

Note that K_i and D modalities can only be applied to $\mathcal{L}_0$ formulae, and also note that primitive propositions — elements of Φ — are *not* formulae of $\mathcal{L}_1$. Let $wff(\mathcal{L}_1)$ be the set of (well-formed) formulae of $\mathcal{L}_1$. The semantics of $\mathcal{L}_1$ are defined via the $\mathcal{L}_1$ satisfaction relation "$\models_{\mathcal{L}_1}$", which holds between tuples of the form $(\kappa_1, \ldots, \kappa_n)$, where $\kappa_i \in KS$ is a knowledge set for agent $i \in Ag$, and formulae of $\mathcal{L}_1$. The rules defining this relation are as follows (we omit the rules for negation and disjunction, as these are trivial):

$$\begin{array}{lll} (\kappa_1, \ldots, \kappa_n) & \models_{\mathcal{L}_1} & K_i\varphi \quad \text{iff } knows(\varphi, \kappa_i) \\ (\kappa_1, \ldots, \kappa_n) & \models_{\mathcal{L}_1} & D\varphi \quad \text{iff } knows(\varphi, \kappa_1 \cap \cdots \cap \kappa_n) \end{array}$$

We refer to tuples of the form $(\kappa_1, \ldots, \kappa_n)$ as $\mathcal{L}_1$ models, for obvious reasons.

Finally, in this paper, we are concerned with *evolving sequences* of knowledge sets. To represent the properties of such sequences, we introduce a temporal logic $\mathcal{L}_2$, which extends $\mathcal{L}_1$ and is in fact defined as a superset of it. $\mathcal{L}_2$ contains two fairly conventional temporal modalities [3]: "$\bigcirc$" (for "next"), and "$\mathcal{U}$" (for "until"), from which the remaining standard connectives of linear discrete temporal logic may be derived. The "$\bigcirc$" connective means "at the next time". Thus $\bigcirc\varphi$ will be satisfied at some time point if φ is satisfied at the *next* time point. The "$\mathcal{U}$" connective means "until". Thus $\varphi \,\mathcal{U}\, \psi$ will be satisfied at some time if ψ is satisfied at that time or some time in the future, and φ is satisfied at all times until the time that ψ is satisfied. The syntax of $\mathcal{L}_2$ is defined by the following grammar:

$$\langle \mathcal{L}_2\text{-}fmla \rangle \quad ::= \quad \langle \mathcal{L}_1\text{-}fmla \rangle \mid \bigcirc\langle \mathcal{L}_2\text{-}fmla \rangle \mid \langle \mathcal{L}_2\text{-}fmla \rangle \,\mathcal{U}\, \langle \mathcal{L}_2\text{-}fmla \rangle \mid \neg\langle \mathcal{L}_2\text{-}fmla \rangle \mid \langle \mathcal{L}_2\text{-}fmla \rangle \vee \langle \mathcal{L}_2\text{-}fmla \rangle$$

The semantics of $\mathcal{L}_2$ are defined with respect to *temporal epistemic models*. Formally, a temporal epistemic model m is simply a function

$$m : \mathbb{N} \rightarrow \underbrace{KS \times \cdots \times KS}_{n \text{ times}}$$

which determines an $\mathcal{L}_1$ model $m(u)$ for every time point $u \in \mathbb{N}$. The semantics of $\mathcal{L}_2$ are given via the $\mathcal{L}_2$ satisfaction relation "$\models_{\mathcal{L}_2}$", which holds between pairs of the form (m, u) (where m is an $\mathcal{L}_2$-model and $u \in \mathbb{N}$ is a temporal index into m), and $\mathcal{L}_2$ formulae. Once again, we only give the semantic rules for non-trivial connectives:

$$\begin{array}{ll} (m,u) \models_{\mathcal{L}_2} \varphi & \text{iff } m(u) \models_{\mathcal{L}_1} \varphi \quad (\text{where } \varphi \in wff(\mathcal{L}_1)) \\ (m,u) \models_{\mathcal{L}_2} \bigcirc\varphi & \text{iff } (m, u+1) \models_{\mathcal{L}_2} \varphi \\ (m,u) \models_{\mathcal{L}_2} \varphi \,\mathcal{U}\, \psi & \text{iff } \exists v \in \mathbb{N} \text{ s.t. } (v \geq u) \text{ and } (m,v) \models_{\mathcal{L}_2} \psi, \\ & \text{and } \forall w \in \mathbb{N}, \text{ if } (u \leq w < v) \\ & \quad \text{then } (m,w) \models_{\mathcal{L}_2} \varphi \end{array}$$

The remaining connectives of linear discrete temporal logic are assumed to introduced as abbreviations as follows:

$$\Diamond\varphi = \mathbf{true}\,\mathcal{U}\,\varphi \qquad \Box\varphi = \neg\Diamond\neg\varphi \qquad \varphi\,\mathcal{W}\,\psi = \varphi\,\mathcal{U}\,\psi \vee \Box\varphi.$$

We now informally consider the meaning of the derived connectives. First, "$\Diamond$" means "either now, or at some time in the future". Thus $\Diamond\varphi$ will be satisfied at some time if either φ is satisfied at that time, or some later time. The "$\Box$" connective means "now, and at all future times". Thus $\Box\varphi$ will be satisfied at some time if φ is satisfied at that time and at all later times. The binary "$\mathcal{W}$" connective means "unless". Thus $\varphi\,\mathcal{W}\,\psi$ will be satisfied at some time if either φ is satisfied until ψ is satisfied, or else φ is always satisfied.

Satisfiability and validity for our three logics are defined in the usual way. We write $\models_{\mathcal{L}_k} \varphi$ to indicate that the $\mathcal{L}_k$ formula φ is valid in $\mathcal{L}_k$. $\mathcal{L}_0$ is simply classical propositional logic, and as such will inherit all the properties of this logic. However, $\mathcal{L}_1$ does not behave exactly like an S5 epistemic logic [4]. In particular, since nested modalities are not permitted in $\mathcal{L}_1$, axioms 4 and 5 are not valid in $\mathcal{L}_1$, and since primitive propositions are not $\mathcal{L}_1$ formulae, axiom T is not valid in $\mathcal{L}_1$. However, we do have the usual K axiom and necessitation rule for K_i modalities:

$$\models_{\mathcal{L}_1} K_i(\varphi \Rightarrow \psi) \Rightarrow ((K_i\varphi) \Rightarrow (K_i\psi))$$
$$\text{If } \models_{\mathcal{L}_0} \varphi \text{ then } \models_{\mathcal{L}_1} K_i\varphi$$

In addition, the D modality has the following properties [4]:

$$\models_{\mathcal{L}_1} D(\varphi \Rightarrow \psi) \Rightarrow ((D\varphi) \Rightarrow (D\psi))$$
$$\models_{\mathcal{L}_1} K_i\varphi \Rightarrow D\varphi$$
$$\text{If } \models_{\mathcal{L}_0} \varphi \text{ then } \models_{\mathcal{L}_1} D\varphi$$

We will not pause to examine the properties of the temporal language $\mathcal{L}_2$, since this has been studied exhaustively elsewhere (see [3] for references).

Finally, we will assume that the notion of *logical consequence* is defined in the standard way for each of our logics. We write $\Gamma \models_{\mathcal{L}_k} \varphi$ to indicate that the $\mathcal{L}_k$ formula φ is an $\mathcal{L}_k$ logical consequence of Γ, where $\Gamma \subseteq wff(\mathcal{L}_k)$.

3 Multi-Agent Systems

The basic idea is to have a collection of programs — agents — which interact with one another by broadcasting messages, where these messages are formulae of $\mathcal{L}_0$. When an agent sends a message φ, the intuitive semantics is that it is asserting the truth of φ. In so doing, the agent is giving other agents in the system some *information*, which they can use to update their own knowledge set. We refer to a system that decides which message to send based upon the agent's internal state as an *agent program*. Abstractly, we view an agent program as a function $pg : KS \rightarrow wff(\mathcal{L}_0)$. Thus, on the basis of an agent's knowledge set, an agent program determines a formula of $\mathcal{L}_0$, which will be the message that the agent sends. Let PG be the set of all such programs. A *multi-agent system* is a tuple of pairs, $((pg_1, \kappa_1^0), \ldots, (pg_n, \kappa_n^0))$, where $pg_i \in PG$ is the program for agent $i \in Ag$, and $\kappa_i^0 \in KS$ is an *initial knowledge set* for agent i. Let Σ be the set of all such multi-agent systems. We use σ (with annotations: $\sigma_1, \sigma', \ldots$) to stand for members of Σ.

We noted above that upon receiving a message, an agent uses the message to update its knowledge set. We model this update process via a *pragmatic interpretation function*:

$$prag : KS \times wff(\mathcal{L}_0) \rightarrow KS$$

where this function is defined as follows:

$$prag(\kappa, \varphi) = \kappa \setminus \{s \mid s \in \kappa \text{ and } s \models_{\mathcal{L}_0} \neg\varphi\}.$$

Thus if an agent receives a message φ, it will remove from its knowledge set all states that are not consistent with φ. The following lemma captures one of the most important property of *prag*.

Lemma 1 *If* $(\ldots, \kappa_i, \ldots)$ *is an* $\mathcal{L}_1$ *model then*

$$(\ldots, prag(\kappa_i, \varphi), \ldots) \models_{\mathcal{L}_1} K_i\varphi$$

for all $\varphi \in wff(\mathcal{L}_0)$.

Proof: Suppose not. Then $\exists s \in prag(\kappa, \varphi)$ such that $s \models_{\mathcal{L}_0} \neg\varphi$. But by construction, s cannot be present in $prag(\kappa, \varphi)$. ∎

We will say a program pg_i is *sincere* if it never generates a message that is not known in the corresponding knowledge set. Formally, a program pg_i is sincere if $pg_i(\kappa) = \varphi$ implies $knows(\varphi, \kappa)$. We will say a system σ is sincere if every program in σ is sincere.

Given the pragmatic interpretation function *prag*, we can define the operation of a multi-agent system. The idea is that every agent i is initially given a knowledge set κ_i^0. It then generates a message $pg_i(\kappa_i^0)$ to send; all other agents do likewise. At the next time step (i.e., time 1), every agent receives all messages that were sent to it at the previous time step (time 0). The conjunction of these messages is used, together with the agent's knowledge set, to generate a new knowledge set, and the process of selecting a message begins again. This model of execution allows us to establish a mapping from multi-agent systems to $\mathcal{L}_2$ models. If $\sigma = ((pg_1, \kappa_1^0), \ldots, (pg_n, \kappa_n^0))$, is a multi-agent system, then the model m_σ of $\mathcal{L}_2$ that represents the execution of σ is defined as follows:

1. $m_\sigma(0) = (\kappa_1^0, \ldots, \kappa_n^0)$ and
2. $\forall u \in \mathbb{N}$ such that $u > 0$, if $m_\sigma(u-1) = (\kappa_1^{u-1}, \ldots, \kappa_n^{u-1})$ then $m_\sigma(u) = (prag(\kappa_1^{u-1}, \chi_{u-1}), \ldots, prag(\kappa_n^{u-1}, \chi_{u-1}))$, where $\chi_{u-1} = pg_1(\kappa_1^{u-1}) \wedge \cdots \wedge pg_n(\kappa_n^{u-1})$.

Thus the formula χ_{u-1} in the second part of this definition is simply the conjunction of all messages sent at time step $u-1$. Note that every agent's knowledge set will *monotonically shrink* as execution of the system proceeds: an agent's knowledge set at time $u+1$ is a subset of its knowledge set at time u. From this observation, it is straightforward to show that the following schema is valid for $\mathcal{L}_2$ models corresponding to systems.

$$\models K_i\varphi \Rightarrow \Box K_i\varphi. \tag{1}$$

It is similarly easy to show that the following schema is valid for $\mathcal{L}_2$ models corresponding to systems.

$$\models D\varphi \Rightarrow \Box D\varphi. \tag{2}$$

Thus distributed knowledge is non-diminishing.

4 Problems

In this section, we formally define problems, and show how multi-agent systems can be used to solve them. Recall that problems that are expressed in terms of logical consequence. We have premises $\varphi_1, \ldots, \varphi_n$, and we wish to know whether some conclusion ψ follows from these premises. Formally, a *problem* is a pair $(\{\varphi_1, \ldots, \varphi_n\}, \psi)$, where $\{\varphi_1, \ldots, \varphi_n\} \subseteq wff(\mathcal{L}_0)$ and $\psi \in wff(\mathcal{L}_0)$. The aim of the problem is to determine whether or not $\{\varphi_1, \ldots, \varphi_n\} \models_{\mathcal{L}_0} \psi$, i.e., whether ψ is an $\mathcal{L}_0$ logical consequence of $\{\varphi_1, \ldots, \varphi_n\}$. If it is indeed the case that $\{\varphi_1, \ldots, \varphi_n\} \models_{\mathcal{L}_0} \psi$, then we say the problem has a positive outcome, otherwise it has a negative outcome. Our main goal in this paper is to investigate multi-agent systems that solve problems of this type.

Intuitively, we say a system *implements* a problem if there is an agent for every premise of the problem, and each agent is initially equipped with *no less* information than its corresponding part of the problem, and *no more* information than the solution of the problem. Formally, a system σ implements a problem $(\{\varphi_1, \ldots, \varphi_n\}, \psi)$ iff $m_\sigma(0) = (\kappa_1^0, \ldots, \kappa_n^0)$ implies that (i) $\kappa_1^0 \subseteq \kappa_{\varphi_i}$, and (ii) $knows(\chi, \kappa_i^0)$ implies $\{\varphi_1, \ldots, \varphi_n\} \models_{\mathcal{L}_0} \chi$. The first condition captures the idea of an agent knowing at least as much as its part of the problem; the second condition captures the idea that an agent can know no more than the whole problem. We can easily establish the following lemma, which relates problems to the knowledge states of agents within a system.

Lemma 2 *If* σ *implements* $(\{\varphi_1, \ldots, \varphi_n\}, \psi)$ *then: (i)* $(m_\sigma, 0) \models_{\mathcal{L}_2} K_i\chi$ *where* $\{\varphi_i\} \models_{\mathcal{L}_0} \chi$*, and (ii)* $(m_\sigma, 0) \models_{\mathcal{L}_2} D\psi$.

Proof: Part (i) follows from the definition of implementing a problem, the definition of knowledge, and straightforward properties of logical consequence; part (ii) follows from (i) using, e.g., axiom (RD2) of [4, p94]. ∎

A system is said to *solve* a problem if eventually, some agent has sufficient information to deduce the conclusion of the problem. In other words, a system σ solves $(\{\varphi_1,\ldots,\varphi_n\},\psi)$ if the *implicit, distributed* knowledge of ψ that the system starts with eventually becomes *explicit* knowledge, possessed by some member of *Ag*. How can it become explicit knowledge? Well, we know the knowledge is there to start with: it just has to be *shared* appropriately, by agents sending each other messages. Formally, system σ is said to *solve* a problem $(\{\varphi_1,\ldots,\varphi_n\},\psi)$ iff it implements it and, moreover $(m_\sigma,0)\models_{\mathcal{L}_2} \Diamond K_i\psi$ for some $i\in Ag$.

Soundness and completeness have natural expressions in our framework. Formally, we say that a system σ which implements a problem $(\{\varphi_1,\ldots,\varphi_n\},\psi)$ is *sound* if

$$(m_\sigma,0)\models_{\mathcal{L}_2}\Diamond K_i\psi \quad \text{implies} \quad \varphi_1\wedge\cdots\wedge\varphi_n\models_{\mathcal{L}_0}\psi$$

and *complete* if

$$\varphi_1\wedge\cdots\wedge\varphi_n\models_{\mathcal{L}_0}\psi \quad \text{implies} \quad (m_\sigma,0)\models_{\mathcal{L}_2}\Diamond K_i\psi.$$

We can make the following general observations about sincerity.

Theorem 1 *If a system is sincere, then: (i) it is sound, and (ii) the following formula schema is true in the model of that system:* $\neg D\varphi\Rightarrow\Box\neg D\varphi$.

Proof: For (i), an easy induction on time points $u\in\mathbb{N}$ shows that if $(m_\sigma,u)\models_{\mathcal{L}_2}K_i\varphi$, then φ must be a logical consequence of the premises. The base case follows from the definition of a system implementing a problem. Then assume that if $(m_\sigma,u)\models_{\mathcal{L}_2}K_i\varphi$, then φ is a logical consequence of the problem premises. For the inductive step, we need to show that $(m_\sigma,u+1)\models_{\mathcal{L}_2}K_i\varphi$ implies φ is a logical consequence of the premises. If $(m_\sigma,u+1)\models_{\mathcal{L}_2}K_i\varphi$, then either $(m_\sigma,u)\models_{\mathcal{L}_2}K_i\varphi$, in which case by the inductive assumption we are done, or else i knows φ as a result of one or more messages it received. But in this case, since σ is sincere, it must be that the messages were sent by agents who knew their content, and so from the inductive assumption, we are done. Part (ii) is straightforward. ∎

An obvious question is whether or not there is a general sound and complete strategy for solving problems in our framework. That is, can we define a general agent program pg_i, such that if used by every agent, it will be guaranteed to solve a problem iff the problem has a solution. As we now demonstrate, such a general complete program does exist. To construct this program pg_i, we proceed as follows. First, we note that there exists an enumeration $seq_1=\chi_0,\chi_1,\chi_2,\ldots$ of $\mathcal{L}_0$ formula, so that every member of $wff(\mathcal{L}_0)$ appears in this sequence eventually [2, p55]. Then, given a knowledge set κ we define another sequence $seq_2=\chi'_1,\chi'_2,\ldots$ by removing from seq_1 every formula χ_u such that not $knows(\chi_u,\kappa)$. Then define the formula χ^* by $\chi^*=\chi'_1\wedge\chi'_2\wedge\cdots$, and let $pg_i(\kappa)=\chi^*$. Intuitively, $pg_i(\kappa)$ will encode *everything* that i knows. We can easily prove the following theorem.

Theorem 2 *A system in which every agent uses this program is sound and complete.*

Proof: Soundness follows from the fact that the program defined in this way is sincere (see Theorem 1). For completeness, observe that at time 0, every agent will send out everything it knows, including its part of the problem, and so by Lemma 1, we will have $(m_\sigma,1)\models_{\mathcal{L}_2}K_i\varphi_1\wedge\cdots\wedge\varphi_n$. Since $\models_{\mathcal{L}_0}\varphi_1\wedge\cdots\wedge\varphi_n\Rightarrow\psi$, we will also have $(m_\sigma,1)\models_{\mathcal{L}_2}K_i\varphi_1\wedge\cdots\wedge\varphi_n\Rightarrow\psi$, and so $(m_\sigma,1)\models_{\mathcal{L}_2}K_i\psi$ and hence $(m_\sigma,0)\models_{\mathcal{L}_2}\Diamond K_i\psi$. ∎

Of course, this example is somewhat unrealistic, in that no *actual* program could ever enumerate seq_1. So while this theorem tells is how agents might be constructed *in principle* to solve problems, it does not offer us much help for building them *in practice*. For this reason, we now consider more practical, implementable agent programs and problem solving strategies. An agent program pg_i must make a decision about the most appropriate message to send based only upon i's knowledge set. How is a program to do this? To see what sort of strategies might work, we will consider *refutation problems*, which as their name suggests, are a class of problems in which the aim is to show some formula is unsatisfiable. Formally, suppose $\varphi_1\wedge\cdots\wedge\varphi_n$ is an $\mathcal{L}_0$ formula that we wish to test for unsatisfiability. Then we know it will be unsatisfiable iff $\models_{\mathcal{L}_0}\varphi_1\wedge\cdots\wedge\varphi_n\Rightarrow\mathbf{false}$. Hence we can test the formula by unsatisfiability by getting a multi-agent system σ to attempt to solve the problem $(\{\varphi_1,\ldots,\varphi_n\},\mathbf{false})$. If $(m_\sigma,0)\models_{\mathcal{L}_2}\Diamond K_i\mathbf{false}$ for some $i\in Ag$, then $\varphi_1\wedge\cdots\wedge\varphi_n$ must be unsatisfiable. The eventual knowledge of **false** by some agent corresponds to the derivation of the empty clause in resolution [7, p130].

Call any problem of the form $(\{\varphi_1,\ldots,\varphi_n\},\mathbf{false})$ a *refutation problem*. A *normal form refutation problem* is one in which each premise φ_i is a disjunction of literals (recall that a literal is a primitive proposition or the negation of a primitive proposition). Finally, a normal form refutation problem is a *Horn* problem if each premise contains at most one positive literal. An example Horn problem (from [6]) is:

$$(\{\underbrace{p}_{\varphi_1},\underbrace{\neg p\vee q\vee\neg r}_{\varphi_2},\underbrace{\neg p\vee\neg q\vee\neg r}_{\varphi_3},\underbrace{\neg p\vee r}_{\varphi_4}\},\underbrace{\mathbf{false}}_{\psi})$$

How can we devise a multi-agent system that will be guaranteed to solve such a problem? One possibility, (based on Fisher's concurrent theorem proving approach [6]) for Horn problems is to give every agent a program pg_i defined as follows:

$$pg_i(\kappa)=\begin{cases}p & \text{if } p\in\Phi,\ knows(p,\kappa)\\ \mathbf{true} & \text{otherwise.}\end{cases}\qquad(3)$$

We will assume as a side condition that agents never send messages that have already been sent. Notice that this program is sincere. In addition, it has an $\mathcal{L}_1$ characterisation:

$$pg_i(\kappa_i)=p \text{ iff } (\ldots,\kappa_i,\ldots)\models_{\mathcal{L}_1}K_ip$$

In this respect, our programs somewhat resemble the *knowledge-based programs* of [4]. We comment further on this relationship in section 5. Despite its obvious simplicity, we can establish the following result.

Theorem 3 *A system in which every agent uses this program is sound and complete for Horn problems.*

Proof: (Outline.) Without loss of generality, we will consider only non-trivial Horn problems. Soundness follows from the fact that the program is sincere. For completeness, first observe that if some Horn clauses $\varphi_1,\ldots,\varphi_n$ are unsatisfiable, then one of them must be a positive literal [7, pp59-60], and in addition, there must be a resolution DAG for these clauses [7, p130]. Our proof is by induction on the depth of this DAG: we show by induction that for all $u\in\mathbb{N}$, if the

clauses have a resolution DAG of depth u, then a system implementing these clauses will solve the problem. The inductive base is where the DAG is of depth 1. Here, the positive literal resolves directly with another clause which must consist solely of the negation of the positive literal. It is easy to see that our system will solve the problem in this case. For the inductive step, we need to show that if the problem has a resolution DAG of depth $u+1$, then the system will solve the problem. The first level in the resolution DAG will involve resolving the positive literal (call it p) with a subset of the other clauses, $\chi_1, \ldots, \chi_k$, to derive their resolvents. Let $\xi_1, \ldots, \xi_l$ be the set of clauses containing the resolvents obtained in this way, together with the clauses that were left unchanged. This set of clauses will be unsatisfiable, and moreover will have a resolution DAG of depth u. In our framework, similar reasoning to the base case shows that p will initially be broadcast. Every agent then removes from their knowledge set any state that does not satisfy p. The key to the proof is to notice that the system at time 1 implements the problem $\xi_1, \ldots, \xi_l$. Since this problem has a resolution DAG of depth u, then by the inductive assumption, it solves it, and we are done. ∎

To illustrate this approach, we dry-run the example given above:

$$(\{p, \neg p \vee q \vee \neg r, \neg p \vee \neg q \vee \neg r, \neg p \vee r\}, \textbf{false}).$$

Initially every agent $i \in \{1, \ldots, 4\}$ has a knowledge set κ_i^0 as follows:

$$\begin{array}{lcl} \kappa_1^0 & = & \{\{p\}, \{p,r\}, \{p,q\}, \{p,q,r\}\} \\ \kappa_2^0 & = & \{\emptyset, \{r\}, \{q\}, \{q,r\}, \{p\}, \{p,q\}, \{p,q,r\}\} \\ \kappa_3^0 & = & \{\emptyset, \{r\}, \{q\}, \{q,r\}, \{p\}, \{p,r\}, \{p,q\}\} \\ \kappa_4^0 & = & \{\emptyset, \{r\}, \{q\}, \{q,r\}, \{p,r\}, \{p,q,r\}\} \end{array}$$

At this point, $pg_1(\kappa_1^0) = p$, and so agent 1 broadcasts p; every other agent broadcasts **true**. Upon receipt of these messages, the state of the system becomes:

$$\begin{array}{lcl} \kappa_1^1 & = & \{\{p\}, \{p,r\}, \{p,q\}, \{p,q,r\}\} \\ \kappa_2^1 & = & \{\{p\}, \{p,q\}, \{p,q,r\}\} \\ \kappa_3^1 & = & \{\{p\}, \{p,r\}, \{p,q\}\} \\ \kappa_4^1 & = & \{\{p,r\}, \{p,q,r\}\} \end{array}$$

We then have $pg_4(\kappa_4^1) = r$, and so agent 4 broadcasts r while every other agent broadcasts **true**. The state of the system is transformed to:

$$\begin{array}{lcl} \kappa_1^2 & = & \{\{p,r\}, \{p,q,r\}\} \\ \kappa_2^2 & = & \{\{p,q,r\}\} \\ \kappa_3^2 & = & \{\{p,r\}\} \\ \kappa_4^2 & = & \{\{p,r\}, \{p,q,r\}\} \end{array}$$

We now have $pg_2(\kappa_2^2) = q$, and so agent 2 broadcasts q; every other agent broadcasts **true**. The state of the system becomes:

$$\begin{array}{lcl} \kappa_1^3 & = & \{\{p,q\}, \{p,q,r\}\} \\ \kappa_2^3 & = & \{\{p,q,r\}\} \\ \kappa_3^3 & = & \emptyset \\ \kappa_4^3 & = & \{\{p,q,r\}\} \end{array}$$

Since $\kappa_3^3 = \emptyset$, we have $(m,3) \models_{\mathcal{L}_3} K_2\textbf{false}$, and the refutation is complete. Extensions to non-Horn problems are not problematic: Fisher demonstrates such techniques in [5], and they can be easily modified for our framework.

5 Conclusions and Related Work

In this paper, we have introduced and investigated a view of distributed problem solving as multi-agent deduction. With this approach, a number of reasoning agents cooperate by exchanging partial results in an attempt to derive a conclusion that could not initially be deduced by any individual agent. We have seen explored a knowledge-theoretic interpretation of this approach, and established some basic results that relate distributed problem solving systems to an epistemic temporal logic.

The work described in this paper builds upon, and is related to that of many other researchers. The most obvious debt is to the work of Fisher and the author, where the basic framework of distributed problem solving as concurrent theorem proving was established [6]. This work, (based in turn upon Fisher's agent-based theorem proving technique [5]), used the Concurrent METATEM multi-agent logic-based programming language to implement a multi-agent planning system. The work in this paper differs from [6] in several respects: it generalises it, gives a precise formal definition of multi-agent problem solving, and finally, uses a knowledge-theoretic approach to analysing systems.

Also closely related is the work of Halpern *et al* on the use of knowledge theory to analyse distributed systems [4]. Halpern and colleagues have studied many aspects of knowledge and distributed knowledge, and in particular, have examined how various states of knowledge can be achieved in message passing systems. However, to the best of my knowledge, no work has been carried out on knowledge-theoretic approaches to problem solving or theorem proving. More recently, attention in the knowledge theory community has shifted to the study of *knowledge based programs*, where agents make decisions about what to do based on their knowledge about the world. Our agent programs are similar to such knowledge-based programs.

REFERENCES

[1] A. H. Bond and L. Gasser, editors. *Readings in Distributed Artificial Intelligence.* Morgan Kaufmann Publishers: San Mateo, CA, 1988.

[2] B. Chellas. *Modal Logic: An Introduction.* Cambridge University Press: Cambridge, England, 1980.

[3] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 996–1072. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1990.

[4] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge*. The MIT Press: Cambridge, MA, 1995.

[5] M. Fisher. An alternative approach to concurrent theorem proving. In J. Geller, H. Kitano, and C. B. Suttner, editors, *Parallel Processing in Artificial Intelligence 3*, pages 209–230. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1997.

[6] M. Fisher and M. Wooldridge. Distributed problem-solving as concurrent theorem proving. In M. Boman and W. Van de Velde, editors, *Multi-Agent Rationality — Proceedings of the Eighth European Workshop on Modelling Autonomous Agents and Multi-Agent Worlds, MAAMAW-97 (LNAI Volume 1237)*, pages 128–140. Springer-Verlag: Berlin, Germany, 1997.

[7] J. Gallier. *Logic for Computer Science: Foundations of Automatic Theorem proving.* John Wiley & Sons, 1987.

[8] L. N. Kanal and V. Kumar, editors. *Search in Artificial Intelligence.* Springer-Verlag: Berlin, Germany, 1988.

[9] R. Kowalski. *Logic for Problem Solving.* Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1979.

[10] V. R. Lesser. A retrospective view of FA/C distributed problem solving. *IEEE Transactions on Systems, Man, and Cybernetics*, 21:1347–1363, 1991.