Verifying space and time requirements for resource-bounded agents

Natasha Alechina
University of Nottingham
Nottingham, UK
nza@cs.nott.ac.uk

Piergiorgio Bertoli
ITC-IRST
Trento, Italy
bertoli@itc.it

Chiara Ghidini
ITC-IRST
Trento, Italy
ghidini@itc.it

Mark Jago
University of Nottingham
Nottingham, UK
mtw@cs.nott.ac.uk

Brian Logan
University of Nottingham
Nottingham, UK
bsl@cs.nott.ac.uk

Luciano Serafini
ITC-IRST
Trento, Italy
luciano.serafini@itc.it

ABSTRACT

We present a novel procedure for automatically verifying the space and time requirements for resource-bounded reasoning agents. We represent agents as a finite state machines in which the states correspond the formulas currently held in the agent's memory and the transitions between states correspond to applying the agent's inference rules. To check whether an agent has enough memory to derive a formula ϕ, we specify the FSM as input to the model-based planner MBP, and check whether the agent has a plan (a choice of memory allocations and inference rule applications), all executions of which lead to states containing ϕ. Our approach is general enough to admit verification of reasoners with any set of inference rules which can be encoded as transitions between FSM states.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent Agents, Multiagent Systems*; I.2.3 [**Artificial Intelligence**]: Deduction and Theorem Proving—*Deduction (e.g., natural, rule-based)*; F.3.1 [**Logics and Meanings of Programs**]: Specifying and Verifying and Reasoning about Programs

General Terms

Design, Verification

Keywords

Resource-bounded agents, MBP

1. INTRODUCTION

The question of how much memory a reasoning agent needs to derive a formula is of considerable theoretical and practical interest. From a theoretical point of view, it is interesting to investigate how the deductive strength of a particular logic changes when only a fixed number of formulas are allowed to be 'active' in a derivation. This is different from the question of bounds on the size proofs [3], since we assume that the whole proof does not need to be held in memory (some intermediate steps may be overwritten).

From a practical point of view, the question of whether an agent will run out of memory or time before achieving its goal(s) is clearly a major concern for the agent developer. As agent tasks become more open ended, the amount of memory required to achieve them becomes harder to predict a priori. For example, the reasoning capabilities of agents assumed by many web service applications is non trivial (e.g., reasoning over complex ontologies or about business processes described by a set of business rules) and the memory requirements correspondingly difficult for the agent developer to determine a priori. At the same time trends towards mobile agents and agents which run on mobile devices such as PDAs and smart phones imply more processor and memory efficient agent designs.

In this paper, we present a novel procedure for automatically verifying the space and time requirements for resource-bounded reasoning agents. Specifically, we address the question: given an agent and a formula ϕ, does the agent have sufficient memory to derive ϕ, and, if it does, what is the length of the shortest derivation within the specified memory bound? In outline, our approach is as follows. We represent a reasoning agent as a finite state machine in which the states correspond the formulas currently held in the reasoner's memory and the transitions between states correspond to applying the reasoning rules. To check whether a reasoner has enough memory to derive a formula ϕ, we specify the FSM as input to the model-based planner MBP [2], and check whether the reasoner has a plan (a choice of memory allocations and inference rule applications), all executions of which lead to states containing ϕ.

Our approach is general enough to admit verification of reasoners with any set of inference rules, provided that those rules can be encoded as transitions between FSM states. To illustrate the generality of our approach, we show how to encode two example reasoners: a forward-chaining rule-based agent of the kind found in many applications employing ontological reasoning and business rules, and a classical propositional reasoner which can derive all classical consequences of its knowledge base given unlimited memory.

AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.

2. FORMAL MODEL

We model resource-bounded agents as finite state machines (FSM) or transition systems. Let the internal language of the agent be some language L (e.g. propositional language). The definition of a transition system is given relative to the following components: the bound n on the agent's memory size; the agent's reasoning rules; the agent's knowledge base $K \subseteq L$; the agent's goal formula $A_G \in L$. The set of all subformulas of K and A_G will be denoted by Ω. We abstract away from the size of the formulas. However, given K and A_G, the maximum size of any formula in the agent's state is fixed.

In the remainder of this section, we first define the language and transition systems for 'definite reasoning' agents and give an example of such an agent. We then introduce a more complex logic for agents that need to maintain a set of epistemic alternatives, such as classical reasoners.

2.1 Definite reasoners

The language of the logic BML^d (for bounded memory logic, definite case) is defined relative to the agent's internal language L. If A is a formula of L, then $\mathrm{B}\,A$ (the agent believes A) is a formula; the language is closed under the boolean connectives and unary modalities EX ('there exists a next state where...') and EF ('there exists a future state where...'). Other boolean connectives are defined in the usual way. We also define $\mathrm{AX}\,\phi$ as $\neg\mathrm{EX}\,\neg\phi$ and $\mathrm{AG}\,\phi$ as $\neg\mathrm{EF}\,\neg\phi$.

A transition system $M = (S, R, V)$ consists of a set of states S, a serial binary relation R on S (transitions between states) and an assignment $V : S \longrightarrow \mathcal{P}(\Omega)$ (assigning to the state the set of formulas the agent believes in that state). Notice that $V(s)$ is not a classical truth assignment, as it might contain complex formulas, e.g., $A \wedge B$, as well as contradictory formulas, e.g., $A \wedge B, \neg B \in V(s)$. To reflect the fact that the agents have bounded memory, we postulate that V can assign at most n formulas to any given state. The transitions which the agent can make depend on the agent's inference rules. In our model, we assume that one of the agent's possible transitions is 'reading' a K formula into its memory or 'active state'. Reading a formula may correspond to reading from flash memory, asking for user input, or reading data from the server over the network.

The definition of a formula being satisfied in $M, s \in S$ is as follows:

$M, s \models \mathrm{B}\,A$ iff $A \in V(s)$

$M, s \models \neg\phi$ iff $M, s \not\models \phi$

$M, s \models \phi \wedge \psi$ iff $M, s \models \phi$ and $M, s \models \psi$

$M, s \models \mathrm{EX}\,\phi$ iff there exists a state t such that $R(s,t)$ and $M, t \models \phi$.

$M, s \models \mathrm{EF}\,\phi$ iff there exists a sequence of states $t_1, \ldots, t_k$ such that for all $i \in \{1, \ldots, k-1\}$, $R(t_i, t_{i+1})$, $t_1 = s$ and $M, t_k \models \phi$

The bound n on the size of the agent's memory is expressed by the following axiom:

B(n) $\mathrm{B}\,A_1 \wedge \ldots \wedge \mathrm{B}\,A_n \rightarrow \neg\mathrm{B}\,A_{n+1}$ where $A_i \neq A_j$ for all $i, j \in \{1, \ldots, n+1\}$ such that $i \neq j$.

We can express that the agent can derive its goal A_G from its knowledge base K as $\mathrm{EF}\,\mathrm{B}\,A_G$ (there is some future state where the agent believes A_G). The fact that a formula is derivable in k steps can be expressed as $\mathrm{EX}^k\mathrm{B}\,A_G$ (where EX^k denotes k applications of the operator EX). Similarly, the fact that an agent needs at least $k+1$ steps to derive a formula A_G can be expressed as $\mathrm{AX}^k\neg\mathrm{B}\,A_G$.

A simple example of a definite reasoner would be an agent which reasons using rules, e.g., ontology rules, or business rules. We assume that agent's knowledge base consists of ground atomic formulas and rules of the form $A_1 \wedge \ldots \wedge A_n \rightarrow B$, where $A_1, \ldots, A_n, B$ are atomic formulas (see, for example, [4]). By generating all possible substitutions of constants occurring in the knowledge base into the rule, we can reduce the knowledge base to a purely propositional set of formulas, consisting of propositional variables and implications of the form $p_1 \wedge \ldots \wedge p_n \rightarrow q$. Then the only rules the agent needs to derive all 'rule-based' consequences are conjunction introduction and modus ponens.

Let $V(s)'$ stand for any subset of $V(s)$ which has cardinality at most $n-1$ and differs from $V(s)$ in at most one formula. The rule-based reasoner has the following transitions:

Read $R(s,t)$ if $V(t) = V(s)' \cup \{A\}$ for some $A \in K$.

AND $R(s,t)$ if $A_1, A_2 \in V(s)$ and $V(t) = V(s)' \cup \{A_1 \wedge A_2\}$.

MP $R(s,t)$ if $A_1 \in V(s)$, $A_1 \rightarrow A_2 \in V(s)$, and $V(t) = V(s)' \cup \{A_2\}$

Reflexivity $R(s,s)$

A complete axiomatisation of this logic is given in [1].

2.2 More general reasoners

It is also interesting to model reasoners which can reason by cases, or in general consider hypothetical states; this means that their transitions do not necessarily follow the logical consequence relation. We also extend the language to express disbelief as well as belief.

Consider a reasoner who believes $\{A \vee B, A \rightarrow C, B \rightarrow C\}$. To derive C, it has to reason by cases: assume A; derive C. Then, assume B; derive C. Hence, it is safe to believe C. However, if the process of assuming A corresponds to a transition to a state where A is believed, the modelling is not 'safe' — the agent's beliefs are not justified by valid inference steps. In the state where it assumes A, the agent should remember that this is just one of the epistemic alternatives, and that in others A is false and B is true.

To deal with such reasoners, we add an extra set of 'epistemic alternatives' or possible worlds to each state. Epistemic alternatives are introduced when the classical reasoner is applying non-deterministic rules, such as elimination of disjunction. Intuitively, a formula is now believed in a state if it is true in all of the epistemic alternatives associated with this state. We express this as $\Box\mathrm{B}\,A$.

The language of the logic BML (for bounded memory logic) extends the language of BML^d by adding extra clauses: if A is a formula of L, then $\overline{\mathrm{B}}\,A$ (the agent disbelieves A) is a formula; if ϕ is a formula, then $\Diamond\phi$ is a formula. We also define $\Box\phi$ as $\neg\Diamond\neg\phi$.

For such general reasoners, we can express that the agent can derive A_G from its knowledge base K as $\mathrm{EF}\,\Box\mathrm{B}\,A_G$ (there is some future state where in all epistemic alternatives the agent believes A_G).

A BML transition system $M = (S, W, R, U, T, F)$ consists of a set of states S, a set of possible worlds or epistemic alternatives W, a binary relation R on S, a function assigning to each state a set of epistemic alternatives $U : S \longrightarrow \mathcal{P}(W)$, and two assignments $T : W \longrightarrow \mathcal{P}(\Omega)$ and $F : W \longrightarrow \mathcal{P}(\Omega)$ which say whether the value of an (internal language) formula in a world is true or false (where, as before Ω is the set of subformulas of K and A_G). As before, to reflect the bound on the agent's memory, we require

$|T(w)| + |F(w)| \leq n$, for any given state w. Moreover, the truth assignments should be consistent, i.e., $T(w) \cap F(w) = \emptyset$. The following truth definitions have been added or modified compared to BML^d. Note that we talk about truth in a world and truth in a state:

$M, w \models \mathrm{B}\, A$ iff $A \in T(w)$

$M, w \models \bar{\mathrm{B}}\, A$ iff $A \in F(w)$

$M, s \models \Diamond\phi$ iff there exists w in $U(s)$, such that $M, w \models \phi$.

The bound n on the size of the agent's memory is expressed by the following axiom:

B'(n) $\Box(\widetilde{\mathrm{B}}\ A_1 \wedge \ldots \wedge \widetilde{\mathrm{B}}\ A_n \rightarrow \neg\ \widetilde{\mathrm{B}}\ A_{n+1})$, where $\widetilde{\mathrm{B}}\ A_i$ stands for either $\mathrm{B}\, A_i$ or $\bar{\mathrm{B}}\, A_i$ and $A_i \neq A_j$ for all $i, j \in \{1, \ldots, n+1\}$ such that $i \neq j$,

The transition relation R between states is defined in terms of expansion relation between epistemic alternatives $\preceq$. Expansion corresponds to applying an inference rule to formulas in the epistemic alternative. Formally, $R(s,t)$ holds if $U(s) = \{w_1, \ldots, w_m\}$, and for some $w_i \in U(s)$, $U(t) = (U(s) \backslash \{w_i\}) \cup \{v : w_i \preceq v\}$.

Note that the classical reasoner agent can construct new formulas in addition to decomposing formulas. We only allow the construction of formulas which are in Ω (the set of subformulas of K and A_G). This does not affect the completeness of agent's rules (since these are the only formulas it may possibly need in the derivation of A_G from K), but allows us to represent it as a finite state machine.

Since the agent can both believe and disbelieve formulas (and its language contains negation), an issue of inconsistent possible worlds arises. An agent *cannot* make a transition to a possible world where the same formula is assigned true and false. All rules therefore have to have a proviso that if $w \preceq v$ then it impossible, for any formula A, to have $A \in T(w)$ and $A \in F(v)$ or vice versa. If the agent starts with inconsistent state, it is allowed to assert any formula from Ω. A full list of transitions the classical reasoner is given in [1].

In [1], we prove that a classical reasoner with unbounded memory can derive A_G from K whenever A_G is a classical consequence of K.

3. VERIFYING REASONING CAPABILITIES

The problem of identifying the existence (and the minimal length) of a deduction for A_G from a knowledge base K for an agent with bounded memory modelled as a transition system M, can be recast as a *planning problem*: find a control strategy for M (a plan) which, starting from any state in K, leads to some state in A_G. In general, M is a *nondeterministic* transition system, since applying a rule may lead to several epistemic alternatives. Thus, we are interested in *strong* plans [2]: tree-structured plans such that their execution leads to the goal, for *every* possible outcome of the actions in the plan.

From the few planners capable of dealing with strong planning for nondeterministic domains we selected MBP, a system which combines effective algorithms with an input language which allows a concise description of transition systems in logical terms. In this section, we provide a high-level description of how the proof existence problem is recast as a planning domain in MBP.

States are identified with assignments $v : \Omega \longrightarrow \{\top, \bot, U\}$ (where U stands for 'undefined'). The transition relation is defined as having as argument the formula to be derived or decomposed, and the formula(s) to be overwritten. For example, the precondition of $\mathbf{Read}(A, B)$ is that A is in KB, and the postcondition is that in the next state, A is true and B is undefined (the agent believes A and does not believe or disbelieve B). If the state is not full, then it is possible that B is a special 'empty' formula (nothing gets overwritten). In some cases a transition may result in more than one formula being overwritten. For example, the precondition for disjunction elimination $\mathbf{elimOr}(A, B_1, B_2)$ is that A is of the form $A_1 \vee A_2$, and that in the current state, $v(A) \neq U$. The postcondition is that in the next state, the value of $v(A_1) \vee v(A_2)$ is equal to the assignment to A in the current state, and B_1, B_2 are assigned U. Note that if in the current state A is assigned $\top$, then the effect of this transition is non-deterministic, since several different values of $v(A_1), v(A_2)$ would satisfy the constraint.

Given this encoding, the planning problem is described by an initial state where $\forall A \in \Omega : v(A) = U$, and by a goal state $v(A_G) = \top$. The constraints on transitions are directly represented in MBP as TRANS constraints over R, allowing for a compact representation of the problem.

MBP implements a number of different search styles. We chose breadth-first backward search which guarantees that the shortest plan is selected. The computational burden imposed by this search style is effectively constrained by the use of symbolic representation techniques that allow a very compact encoding, and an efficient handling of extremely large state sets; details can be found in [2].

Experimental results can be found in [1].

4. CONCLUSIONS AND FUTURE WORK

In this paper, we have attempted to take seriously the idea that reasoning is a process which requires memory and time. While the temporal aspect of reasoning has been studied before, we believe that our treatment of the memory aspect is novel. We have proposed a new kind of epistemic logic where both resources are explicitly modelled. The logic is interpreted on state transition systems, where the agent's state can contain only a fixed finite number of formulas (beliefs), and transitions correspond to application of inference rules by the agent. By specifying the state transition system as an input to the MBP planner, we can automatically verify the lower bounds on space and time required by the agent to derive a certain formula.

In future work, we plan to model multi-agent systems, including agents which are capable of reasoning about other agents' beliefs and their reasoning in epistemic logic. We also would like to model reasoners in description logics, since verifying agents which do ontological reasoning is one of our prime motivations.

Acknowledgements This work was supported by the Royal Society UK-Italy Joint Project grant 'Model-checking resource-bounded agents'.

5. REFERENCES

[1] N. Alechina, P. Bertoli, C. Ghidini, M. Jago, B. Logan, and L. Serafini. Verifying space and time requirements for resource-bounded agents. Technical Report T05-10-03, ITC-irst, Trento, Italy, 2005.

[2] A. Cimatti, M. Pistore, M. Roveri, and P. Traverso. Weak, Strong, and Strong Cyclic Planning via Symbolic Model Checking. *Artificial Intelligence Journal*, 147(1,2):35–84, July 2003.

[3] A. Haken. The intractability of resolution. *Journal of Theoretical Computer Science*, 39(2-3):297–308, Aug. 1985.

[4] I. Horrocks and P. F. Patel-Schneider. A proposal for an OWL rules language. In *Proceedings of the 13th international conference on World Wide Web, WWW 2004*, pages 723–731. ACM, 2004.