Model-checking space and time requirements for resource-bounded agents

Natasha Alechina[1], Piergiorgio Bertoli[2], Chiara Ghidini[2], Mark Jago[1], Brian Logan[1], and Luciano Serafini[2]

[1] School of Computer Science
University of Nottingham
Nottingham, UK
{nza,mtw,bsl}@cs.nott.ac.uk
[2] ITC-IRST, Trento, Italy
{bertoli,ghidini,luciano.serafini}@itc.it

Abstract. The *effective reasoning capability* of an agent can be defined as its capability to infer, within a given space and time bound, facts that are logical consequences of its knowledge base. In this paper we show how to determine the effective reasoning capability of an agent with limited memory by encoding the agent as a transition system and automatically verifying whether a state where the agent believes a certain conclusion is reachable from the start state. We present experimental results using the Model Based Planner (MBP) which illustrates how the length of the deduction varies for different memory sizes.

1 Introduction

Consider an agent that has a finite knowledge base and some rules of inference which allow it to derive new information from its knowledge base. It is intuitively clear that some derivations require more memory than others (e.g., to store intermediate results), and that two agents with the same knowledge base and the same set of inference rules, but with different amounts of memory, may not be able to derive the same formulas.

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. 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 (e.g., the Micro-FIPA-OS [19] and JADE-LEAP [5] platforms). Such devices typically have a relatively small amount of physical memory (and no virtual memory), which must be shared between the OS, the agent platform and other applications running on the device. While increased bandwidth and more powerful handheld devices will undoubtedly become available, the rapid growth in, e.g., the number and complexity of ontologies, seems likely to outstrip any increases in hardware capabilities, at least for the foreseeable future.

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. 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 classical propositional reasoner which can derive all classical consequences of its knowledge base given unlimited memory, and a forward-chaining rule-based agent of the kind found in many applications employing ontological reasoning and business 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 [7], 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. Using a simple business rules example, we show how MBP can be used to automatically verify the existence of a derivation, and present experimental results which illustrate how the length of the deduction varies for different memory sizes.

The remainder of the paper is organised as follows. In section 2 we introduce our model of the agent's memory and give some examples of the kinds of properties we wish to verify. In section 3, we present our formal model of a resource bounded agent and show how to model two example agents, a simple agent that reasons using rules, and a classical reasoner capable of deriving any classical consequence of its knowledge base. In section 4 we briefly introduce the MBP model-based planner and explain how it is used to verify the memory requirements of a resource-bounded reasoner. In section 5 we present a simple example to illustrate the effects of memory limitations on a rule-based reasoning agent and give results from MBP illustrating how the length of deduction varies for different memory sizes. In section 6 we briefly describe related work before concluding in section 7.

2 Memory bounds

Consider an agent running on a small device like a mobile phone, a simple PDA, or even a smaller device like a node of a sensor network. The agent has a pool of potentially available information stored in a Knowledge Base (K)[3] and a fixed set of reasoning

[3] The information could be stored in a remote database or in a persistent memory like a flash card or obtained in input from a user. In this paper we abstract from these aspects and say only that information is potentially available in a knowledge base K.

rules. Using information from the knowledge base and the inference rules, the agent can infer new formulas. We assume that the knowledge base is too large to fit into the agent's memory, and the agent can store at most n formulas from K in memory at any given time. Loading new information from the KB when the agent's memory is full overwrites some of the information currently in memory. For example, a location-aware device which advises a traveller about local amenities and tourist attractions cannot load an entire database of attractions and ontological definitions in memory when computing a recommendation, and will have to manage the subset of formulas from K which are in memory and available for inference. Given this resource bound, which we call, 'memory of size n', the properties we are interested in verifying are of the form: can a formula ϕ be derived with a memory of size n?; what is the minimum amount of memory required to derive ϕ?; is there a relation between memory size and the number of steps required to derive ϕ? what is the minimum amount of memory required to derive ϕ with the shortest derivation?

To illustrate the impact of memory bounds in the reasoning process, consider an agent with a knowledge base K composed of the following formulas:

$$A, A \rightarrow B, B \rightarrow C, C \rightarrow D. \tag{1}$$

If the only inference rule the agent uses is modus ponens, it will require a memory of at least size 2 to derive D:

1. read A (memory contains $\{A\}$)
2. read $A \rightarrow B$ (memory contains $\{A,\ A \rightarrow B\}$)
3. apply modus ponens and store B, overwrite A (memory contains $\{A \rightarrow B,\ B\}$)
4. read $B \rightarrow C$, overwrite $A \rightarrow B$ (memory contains $\{B,\ B \rightarrow C\}$)
... $\ldots$
n. until we apply all the rules and conclude D.

The deduction above requires only two formulas in memory at any given time as we can overwrite the antecedent of an implication with the result of applying modus ponens, load the next implication, apply modus ponens, and store the new result. Notice that after adding new implications, say $E \rightarrow F$, $F \rightarrow G$, we still need only two formulas in memory to derive G. Thus memory requirements do not necessarily depend upon the number of formulas used in the derivation. However, if K contains the following formulas

$$A, A \rightarrow B, A \wedge B \rightarrow C, B \wedge C \rightarrow D \tag{2}$$

and the agent reasons using the inference rules modus ponens (MP) and conjunction introduction ($\wedge_I$), then the derivation requires storing at least 3 formulas in memory at any given time. Notice that the two knowledge bases (1) and (2) are logically equivalent. Thus memory requirements can change for logically equivalent knowledge bases. Also, it can be shown that adding conjunction elimination to the set of rules allows the agent to derive D with only 2 formulas in memory. Thus, memory requirements also depend upon the inference rules available to the agent.

In summary, there is a trade-off between space and time requirements, and the memory required for a derivation will depend on both K and the agent's inference rules.

Given a procedure for determining how much memory a given derivation requires (and how much time it takes) for particular inference rules and K, an agent developer can ensure that an agent has sufficient memory for a particular task, or, conversely engineer a K which will allow an agent with particular inference capabilities and memory size to derive a given formula.

3 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:

1. the bound n on the agent's memory size
2. the agent's reasoning rules
3. the agent's knowledge base $K \subseteq L$
4. 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, the maximal size of any formula which the agent's state has information about, will be fixed.

In the remainder of this section, we first define the language and transition systems for 'definite reasoning' agents, which never do reasoning by cases or assumption-based reasoning, and give an example of such an agent (rule-based agent). We then introduce a more complex logic for agents that need to maintain a set of epistemic alternatives, and give an example of such an agent (classical reasoner).

3.1 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. Well formed formulas (w.f.f.) are defined as follows:

- If A is a formula of L, then $\mathrm{B}\, A$ (the agent believes A) is a w.f.f.
- If ϕ is a w.f.f., then $\neg\phi$, $\mathrm{EX}\,\phi$ ('in one of the successor states, ϕ') and $\mathrm{EF}\,\phi$ ('in some future state, ϕ') are w.f.f.
- If ϕ_1 and ϕ_2 are w.f.f., then $\phi_1 \wedge \phi_2$ is a w.f.f.

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 a 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 $M, t_k \models \phi$

Let **M** be a class of models (for example, all models with the same knowledge base and the same transition rules). A formula is **M**-satisfiable if it is true in some state in some model in **M**. A formula is **M**-valid if it is true in every state in every model in **M**. The definition of logical consequence is standard.

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

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

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 $\mathrm{AX}^k \neg \mathrm{B}\, A_G$.

3.2 Rule-based reasoners

In this section we present a simple example of 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, [14]). An example of such rule would be

$$Parent(x, y) \wedge Brother(y, z) \rightarrow Uncle(x, z)$$

Essentially, such agents can only reason by a single inference rule:

$$\frac{A_1(\bar{a}), \ldots, A_n(\bar{a}) \quad \forall \bar{x}(A_1(\bar{x}) \wedge \ldots \wedge A_n(\bar{x}) \rightarrow B(\bar{x}))}{B(\bar{a})}$$

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 $p_1 \wedge \ldots \wedge$

$p_n \to q$. Then the only rules the agent needs to derive all 'rule-based' consequences are conjunction introduction $\wedge_I$ and modus ponens MP:

$$\frac{A_1,\ A_2}{A_1 \wedge A_2}\wedge_I \qquad \frac{A_1,\ A_1 \to A_2}{A_2}MP$$

We show how to represent this reasoner as an FSM. Let $V'(s)$ be any subset of $V(s)$ which differs from $V(s)$ in at most one formula and has cardinality at most $n-1$. 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 $A_1 \in V(s)$, $A_1 \to A_2 \in V(s)$, and $V(t) = V(s)' \cup \{A_2\}$.
Reflexivity $R(s,s)$

For technical convenience (we will discuss a class of models without this assumption later in this section), we also allow (but not require) 'forgetting' transitions of the form $R(s,t)$, where $V(t) = V'(s)$.

Notice that the definition of $V'(s)$ guarantees that after each transition $R(s,t)$, the memory bound is satisfied by $V(t)$, i.e., $|V(t)| \leq n$.

A formula A_G is derivable from K using only modus ponens and conjunction introduction with memory of size n if, and only if, $M_{K,A_G}, start \models \mathrm{EF}\,\mathrm{B}\,A_G$, where M_{K,A_G} is a rule-based transition model where states are assigned only formulas which are subformulas of K and A_G, $V(s)$ for any s contains at most n formulas, and $V(start) = \emptyset$. Indeed, a derivation of A_G from K using only the allowed rules and at most n formulas in memory corresponds to a branch in a state transition system described above from an empty state to a state containing A_G; and conversely, such a branch can be converted into a derivation of A_G from K. Similarly, A_G is derivable from K in k steps iff $M_{K,A_G}, start \models \mathrm{EX}^k\mathrm{B}\,A_G$.

The logical axioms corresponding to the rule-based reasoner's transition rules are as follows (we assume $n \geq 1$ for **A1**):

A1 $\mathrm{EX}\,\mathrm{B}\,A$ for $A \in K$
A2 $\mathrm{B}\,A_1 \wedge \mathrm{B}\,A_2 \to \mathrm{EX}\,\mathrm{B}\,(A_1 \wedge A_2)$
A3 $\mathrm{B}\,A_1 \wedge \mathrm{B}\,(A_1 \to A_2) \to \mathrm{EX}\,\mathrm{B}\,A_2$

Finally, we need to express that only transitions which are made according to the rules are possible, and that in each transition at most one new formula is added and at most one formula is overwritten.

A4 $\mathrm{EX}\,(\mathrm{B}\,A_1 \wedge \mathrm{B}\,A_2) \to \mathrm{B}\,A_1 \vee \mathrm{B}\,A_2$
A5 $\mathrm{EX}\,(\neg\mathrm{B}\,A_1 \wedge \neg\mathrm{B}\,A_2) \to \neg\mathrm{B}\,A_1 \vee \neg\mathrm{B}\,A_2$
A6 $\mathrm{EX}\,\mathrm{B}\,(A_1 \wedge A_2) \to \mathrm{B}\,(A_1 \wedge A_2) \vee (\mathrm{B}\,A_1 \wedge \mathrm{B}\,A_2)$
A7 $\mathrm{EX}\,\mathrm{B}\,A_2 \to \mathrm{B}\,A_2 \vee \bigvee_{A_1 \to A_2 \in K}(\mathrm{B}\,(A_1 \to A_2) \wedge \mathrm{B}\,A_1)$ for $A_2 \notin K$ and $A_2 \neq B \wedge C$

Note that the only axiom schema which depends on K is **A7**. Let $ML(K, n, \mathrm{EX})$ be the logic defined by the set of axiom schemata **A1 - A7, B(n)**, together with the classical and modal axioms for EX:

Cl tautologies of classical logic
K $\text{AX}(\phi \rightarrow \psi) \rightarrow (\text{AX}\,\phi \rightarrow \text{AX}\,\psi)$
T $\phi \rightarrow \text{EX}\,\phi$
MP $\vdash \phi, \vdash \phi \rightarrow \psi \Rightarrow \vdash \psi$
N $\vdash \phi \Rightarrow \vdash \text{AX}\,\phi$

Let $\mathbf{M}(K, n)$ stand for the class of models where the knowledge base is K, the memory size is n, and the only possible transitions are defined by the transition rules above. We then have the following completeness result.

Theorem 1. *$ML(K, n, \text{EX})$ is sound and strongly complete with respect to $\mathbf{M}(K, n)$.*

We omit the proof due to the lack of space; it can be found in [2].

3.3 More general reasoners

In this section, we 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. 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 $\Box \text{B}\, A$.

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 $\bar{\text{B}}\, A$ (the agent disbelieves A) is a w.f.f.
- If ϕ is a w.f.f., then $\Diamond\phi$ is a w.f.f.

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 $\text{EF}\,\Box\text{B}\,A_G$ (there is some future state where in all epistemic alternatives the agent believes A_G).

A BML transition system $M = (S, W, R, Y, T, F)$ 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 $Y : S \longrightarrow \mathcal{P}(W)$, 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 $Y(s)$, such that $M, w \models \phi$.

The bound n on the size of the agent's memory is expressed by the following axiom (which replaces **B(n)** defined for BML^d):

$\mathbf{B(n)}'$ $\Box(\widetilde{\mathrm{B}}\ A_1 \wedge \ldots \wedge \widetilde{\mathrm{B}}\ A_n \rightarrow \neg\, \widetilde{\mathrm{B}}\ A_{n+1})$, 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 $i \neq j$.

3.4 Classical reasoners

In this section we present a simple example of a classical reasoner, which, given unlimited memory, is capable of deriving any classical consequence of its knowledge base.

Epistemic alternatives are introduced when the classical reasoner applies non-deterministic rules, such as disjunction elimination. Suppose, for example, that the agent has $A \vee B$ in its knowledge base and starts in a state s_0, which has a single epistemic alternative w_0 with $T(w_0) = F(w_0) = \emptyset$. The agent can read $A \vee B$ and transit to a state s_1 with a single epistemic alternative w_1, such that $A \vee B \in T(w_1)$. Now the agent applies a non-deterministic rule for disjunction; it may assume that both disjuncts are true, or A is true and B is false, or vice versa. Formally, this means that the agent transits to a state s_2 where the epistemic alternatives are:

1. w_{11} with $A, B \in T(w_{11})$,
2. w_{12} with $A \in T(w_{12})$ and $B \in F(w_{12})$,
3. w_{13} with $B \in T(w_{12})$ and $A \in F(w_{12})$.

Note that the classical reasoner cannot derive A from $A \vee B$ in the sense of our criterion of EF $\Box\mathrm{B}\, A$ being true: A is true in w_{11} and w_{12}, but false in w_{13}, so s_2 does not satisfy $\Box\mathrm{B}\, A$.

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; in the example above, w_1 is expanded (by applying the rule of disjunction elimination) to w_{11}, w_{12}, w_{13}. Formally, $R(s, t)$ holds if $Y(s) = \{w_1, \ldots, w_m\}$, and for some $w_i \in Y(s)$, $Y(t) = (Y(s) \backslash \{w_i\}) \cup \{v : w_i \preceq v\}$.

Before we define the expansion relation, we need a few preliminary definitions and comments. 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 to 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:

Recall $w \preceq v$ and $A \in T(v) \Rightarrow A \notin F(w)$, and $w \preceq v$ and $A \in F(v) \Rightarrow A \notin T(w)$

Here is a list of possible types of transitions:

Read $w \preceq v$ if for some formula $A \in K$, $A \in T(v)$, and otherwise $T(v), F(v)$ contain the same formulas as $T(w), F(w)$, apart from possibly omitting one (overwritten) formula. Observe that w can be expanded by the **Read** transition in as many ways as there are formulas in K, and choices for overwriting a formula in $T(w) \cup F(w)$ (including a choice to overwrite nothing). In the modelling section, these two formulas (a formula added and a formula overwritten) are made explicit parameters in defining the transition.

Split $w \preceq v_1$ and $w \preceq v_2$ if for some formula $A \in \Omega$ with $A \notin T(w) \cup F(w)$, $A \in T(v_1)$, $A \in F(v_2)$, and otherwise the truth assignment in v_1, v_2 is the same as in w, with at most one formula in each world being overwritten, and **Recall** is satisfied. This transition rule enables the agent to do reasoning by cases, and is equivalent to having $A \vee \neg A$ as an axiom, for every $A \in \Omega$.

ExContradictio $w \preceq v$ if for some A, $A \in T(w)$ and $\neg A \in T(w)$, or $A \in F(w)$ and $\neg A \in F(w)$, and $T(v)$ contains A_G.

makeNot $w \preceq v$ if for some $\neg A \in \Omega$, $A \in T(w) \cup F(w)$, and $\neg A \in T(v) \cup F(v)$ with the opposite sign, otherwise the truth assignment in v is the same as in w (with at most one formula possibly overwritten), and **Recall** is satisfied.

elimNot $w \preceq v$ if for some $\neg A \in \Omega$, $\neg A \in T(w) \cup F(w)$, and $A \in T(v) \cup F(v)$ with the opposite sign, otherwise the truth assignment in v is the same as in w (with at most one formula possibly overwritten), and **Recall** is satisfied.

makeAnd $w \preceq v$ if for some $A_1 \wedge A_2 \in \Omega$, $A_1, A_2 \in T(w) \cup F(w)$ and $A_1 \wedge A_2 \in T(v) \cup F(v)$, so that the value of $A_1 \wedge A_2$ in v is the logical 'and' of the values of A_1, A_2 in w, otherwise the truth assignment in v is the same as in w (with at most one formula possibly overwritten), and **Recall** is satisfied.

elimAnd $w \preceq v$ if $A_1 \wedge A_2 \in T(w) \cup F(w)$, $A_1, A_2 \in T(v) \cup F(v)$, so that the logical 'and' of the truth value of A_1 and A_2 in v equals to the value of $A_1 \wedge A_2$ in w, otherwise the truth assignment in v is the same as in w (with at most *two* formulas possibly overwritten), and **Recall** is satisfied. If the conjunction is true in w, there is only one possible truth assignment to the conjuncts in v, but if it is false, then w can be expanded by this rule to worlds where one of the conjuncts is true and another one false, or both false.

Transition rules for other connectives are defined in similar fashion.

Theorem 2. *A classical reasoner with unbounded memory can derive A_G from K whenever A_G is a classical consequence of K.*

Proof. Let A_G be a classical consequence of K. If K is inconsistent, we use **ExContradictio** to derive A_G. If K is consistent, the strategy for deriving A_G is as follows. The reasoner does not overwrite any formulas. It reads all formulas from K and decomposes them down to all possible assignments to propositional variables in K. If variables of A_G are a subset of the variables of K, then each branch in the previous execution can be continued with a successful composition of A_G (since every assignment satisfying K satisfies A_G. Else let $Var(A_G)\backslash Var(K) = \{q_1, \ldots, q_m\}$. Then, continue each branch of the previous derivation with m splits on each of q_i. This will generate all possible assignments to $Var(K) \cup \{q_1, \ldots, q_m\}$ which make K true. By assumption, each of them makes A_G true, so again on each branch A_G can be successfully assembled.

4 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) such that, starting from any state in K, it leads to some state in A_G. The plan is the proof of A_G.

In general, M is a *nondeterministic* transition system, since applying a rule may lead to several epistemic alternatives, as shown e.g. in Sec. 3.4 for the case of disjunction elimination. Thus, we are interested in *strong* plans [7]: tree-structured plans such that their execution leads to the goal, for *every* possible outcome of the actions in the plan.

Among the few planners capable to deal with strong planning for nondeterministic domains, we selected MBP, a system coupling 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 the way the proof existence problem is recast as a planning domain in MBP. We take as reference the classical reasoner, leaving the simpler case of rule-based reasoning to the reader. For reasons of space, we will omit the encodings of the rules associated to disjunction and implication, which are analogous to the one for conjunction.

In the following, we partition Ω into the subsets Ω_0, $\Omega_\neg$, $\Omega_\vee$, $\Omega_\wedge$, $\Omega_\rightarrow$ which contain respectively atomic formulae, and formulae whose top-level connective is a negation, disjunction, conjunction or implication. Moreover we define the functions $l(\cdot)$ and $r(\cdot)$ which return the left/right parts of non-atomic formulas. We omit their trivial definition, and we take the convention that $l(\neg\phi) = \phi$.

The core of the encoding consists in representing the state transition system described in Section 3 as a planning domain. Formally, a planning domain is a triple (S, Act, R), where S are the states of the domain, Act is a set of actions, and $R \subseteq S \times Act \times S$ is the transition relation, describing the outcomes of the action execution; an action is executable over a state s iff $\exists(s, \alpha, s') \in R$. Our mapping views actions as deduction rules and domain states as epistemic states of the agent. In a planning domain, the state is represented by means of a set of *state variables*. In our case, the set V will be composed of $|\Omega|$ three-valued state variables. We will denote with $V(\phi)$ the value of the variable associated to ϕ. $V(\phi)$ corresponds to the believed value of ϕ ($\top$ or $\bot$), or indicates that nothing is believed about it (U), representing the T, F assignments of the transition system for BML. The memory bound condition is enforced by a constraint $\Psi_{\leq n}$ on R, of the form $|\{A : V(A) \neq U\}| \leq n$, directly represented in MBP as a $TRANS\ \Psi_{\leq n}$ construct.

The actions of the domain represent every possible instance of the deduction rules (Read, Split, etc.) over the formulas in Ω. Such instantiation must also explicitly consider, for a given action, every possible choice of the formula(s) to be overwritten by the newly produced formula(s). As such, actions feature one argument in Ω representing the formula to be read, split, or composed, and one or two additional arguments in $\Omega' = \Omega \cup \{A_0\}$, indicating the formula(s) to be overwritten, and the fictitious formula A_0 if no rewriting occurs. This defines the range of the action variable α in the planning domain:

$$\alpha \in \bigcup_{\substack{A \in K \\ B \in \Omega'}} \mathbf{Read}(A,B) \cup \bigcup_{\substack{A \in \Omega \\ B \in \Omega'}} \mathbf{Split}(A,B) \cup \bigcup_{\substack{A \in \Omega \\ B \in \Omega'}} \mathbf{ExC}(A,B) \cup \bigcup_{\substack{A \in \Omega_\neg \\ B \in \Omega'}} \mathbf{makeNot}(A,B) \cup$$

$$\bigcup_{\substack{A \in \Omega_\neg \\ B \in \Omega'}} \mathbf{elimNot}(A,B) \cup \bigcup_{\substack{A \in \Omega_\wedge \\ B \in \Omega'}} \mathbf{makeAnd}(A,B) \cup \bigcup_{\substack{A \in \Omega_\wedge \\ B_1 \neq B_2 \\ B_1, B_2 \in \Omega'}} \mathbf{elimAnd}(A, B_1, B_2)$$

The executability preconditions and the effects of the actions are encoded in MBP as an implicitly conjoined set of constraints over the transition relation, again of the form $TRANS\ \Psi$.

The executability preconditions correspond to the constraints on the current world in the transition rules in Section 3:

$$\begin{aligned}
&\alpha = \mathbf{Read}(A,B) \to A \in K \\
&\alpha = \mathbf{ExC}(A,B) \to U \neq V(A) = V(l(A)) \\
&\alpha = \mathbf{makeNot}(A,B) \to V(l(A)) \neq U \\
&\alpha = \mathbf{elimNot}(A,B) \to V(A) \neq U \\
&\alpha = \mathbf{makeAnd}(A,B) \to V(l(A)) \neq U \wedge V(r(A)) \neq U \\
&\alpha = \mathbf{elimAnd}(A,B_1,B_2) \to V(A) \neq U
\end{aligned}$$

The (possibly nondeterministic) effects of an action are represented by partitioning the effects over the formula(s) read or built by the rule, and those over the formula(s) that are possibly overwritten by the result(s) of its application. The former are written in terms of the values V must attain for the affected formula(s) after the action execution (i.e. at the next step, denoted with X), constrained by the current values of V, according to the definitions in Section 3.

$$\begin{aligned}
&\alpha = \mathbf{Read}(A,B) \to X(V(A) = \top) \\
&\alpha = \mathbf{Split}(A,B) \to X(V(A) \in \{\top, \bot\}) \\
&\alpha = \mathbf{ExC}(A,B) \to X(V(A_G) = \top) \\
&\alpha = \mathbf{makeNot}(A,B) \to X(V(A)) = \neg(V(l(A))) \\
&\alpha = \mathbf{elimNot}(A,B) \to X(V(l(A))) = \neg(V(A)) \\
&\alpha = \mathbf{makeAnd}(A,B) \to X(V(A)) = V(l(A)) \wedge V(r(A)) \\
&\alpha = \mathbf{elimAnd}(A,B_1,B_2) \to V(A) = X(V(l(A)) \wedge V(r(A)))
\end{aligned}$$

The following constraints ensure that overwritten formulas become undefined:

$\alpha = \mathbf{Read}(A, B) \wedge B \notin \{A_0, A\} \rightarrow X(V(B) = U)$
$\alpha = \mathbf{Split}(A, B) \wedge B \notin \{A_0, A\} \rightarrow X(V(B) = U)$
$\alpha = \mathbf{ExC}(A, B) \wedge B \notin \{A_0, A_G\} \rightarrow X(V(B) = U)$
$\alpha = \mathbf{makeNot}(A, B) \wedge B \notin \{A_0, A\} \rightarrow X(V(B) = U)$
$\alpha = \mathbf{elimNot}(A, B) \wedge B \notin \{A_0, l(A)\} \rightarrow X(V(B) = U)$
$\alpha = \mathbf{makeAnd}(A, B) \wedge B \notin \{A_0, A\} \rightarrow X(V(B) = U)$
$\alpha = \mathbf{elimAnd}(A, B_1, B_2) \wedge B_1, B_2 \notin \{A_0, l(A), r(A)\} \rightarrow X(V(B_1) = V(B_2) = U)$

The constraints above must be conjoined with those representing the **Recall** proviso, and the provisos on the inertiality of the values of non-affected formulas. **Recall** is expressed by adding, for each $A \in \Omega$, two constraints of the form $V(A) = \top \rightarrow X(V(A)) \neq \bot$ and $V(A) = \bot \rightarrow X(V(A)) \neq \top$. Inertiality is expressed by adding constraints stating explicitly that unless a formula is overwritten or produced, it does not change its value, e.g.:

$$\alpha = \mathbf{makeAnd}(A, B) \wedge A' \notin \{A, B\} \rightarrow X(V(A')) = V(A')$$

Given the encoding above, the planning problem is described by an initial state where $\forall A \in \Omega : V(A) = U$, and by a goal state $V(A_G) = \top$.

MBP implements many possible search styles. We chose breadth-first backward search which guarantees that the shortest plan is selected. The computational burden imposed by such a 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 at once; details can be found in [7].

5 Experiments

We present a simple example to illustrate the effect of memory size on the minimum length of a derivation. Consider the set of rules

$$A \wedge B \rightarrow H \qquad B \wedge I \wedge E \rightarrow L \qquad F \wedge G \rightarrow M$$
$$A \wedge B \rightarrow C \qquad D \wedge A \wedge H \rightarrow I \qquad I \wedge L \wedge M \rightarrow N$$

which may form part of a larger knowledge base. Suppose a designer of a system which uses a knowledge base containing these rules wishes to verify, e.g., that from the following basic facts $\{B, D, E, F, G\}$ an agent running on a PDA with a memory of size n can infer $N \wedge C$. In addition, the designer may be interested in how increases in memory size affect the number steps required for the derivation, e.g., if they wish to trade memory for response time.

Figure 1 shows the length of the shortest deduction of the formula $N \wedge C$ for different memory sizes as determined by the MBP planner. Deriving the target formula requires a memory of at least size 3. For memory size of 1 and 2 the system quickly determines that there are no possible derivations of the target formula. Let us focus on the

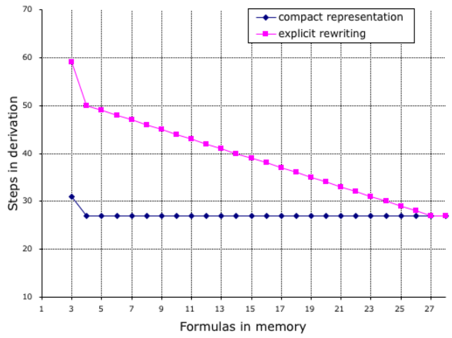

Fig. 1. Running the example

lower curve. With 3 memory cells the deduction requires 31 steps. With a memory of size 4, the number of steps in the deduction drops to 27. This is because the fourth cell is used to store an intermediate result which is used more than once in the derivation and does not need to be recomputed, thus shortening the inference process. In this example further increases in the amount of memory do not result in further reductions in the length of derivation. These results do not consider explicitly the action of overwriting a memory location, that is, steps in the derivation consist either of the application of an inference rule or reading a formula from K. In computing the length of a deduction we may also want to explicitly consider the action of over-writing a memory location (we can think of this step as choosing which location in memory to over-write). The upper curve in Figure 1 shows the length of the derivations including these extra steps. With a memory of size 3, the number of steps in the derivation is 59 (31 steps + 28 over-write operations). With a memory of size 4 this drops to 50 steps (27 steps + 23 over-write operations). As can be seen, the number of times a cell in over-written continues to drop with increasing memory size, until with a memory of size 27, when we can store all the subformulas used in the derivation in memory, the length of the derivation is the same as in the previous case.

6 Related work

Our work is related to other work on logics of knowledge and belief, for example [11]. Much of this work assumes that the agent's knowledge is deductively closed, and therefore does not try to model the time and space restrictions on an agent's ability to derive the consequences of its beliefs. There is a growing body of work in which the agent's deduction steps are explicitly modelled in the logic, for example [10, 8, 3, 1]. These approaches make it possible to model the time it takes the agent to arrive at a certain conclusion, but not the space required. A different kind of limitation on the depth of belief reasoning allowed is studied in [13]. Limitations on memory are considered in fewer approaches; for example, in work on the logic of games [20], where an agent with limited memory can base its strategy only on a limited portion of the game's history, and in some of the work on step logic [9], which considers both the time and space limitations on the agent's knowledge. Step logic makes use of the notion of a *step* in reasoning. Given a set of formulas X and a set of inference rules I, an agent performs a step of reasoning by adding the consequents of any applicable inference rule in I to X. If a formula ϕ had been derived in this way at step t, it is said to be a t-theorem. [9] address the issue of the increasing number of t-theorems at each step, which require a larger and larger memory size. However, rather than attempting to verify the space required to solve a given problem, [9] are concerned with restricting the size of short term memory to isolate any possible contradictions, thereby avoiding the problem of *swamping*: deriving all possible consequences from a contradiction. The emphasis on perfect rationality in AI was challenged by Russell in [18] in favour of bounded optimality, (optimality relative to the time and space bounds on the device the agent program is running on).

The problem of formal verification of multi agent systems has lead to a growing body of work, especially in the area of multi agent model checking [4, 16]. The existing work, however, is mainly focused on logically omniscient agents, that is, agents who instantaneously believe all the logical consequences of their basic beliefs, and no time and space limitations are taken into account.

The connection between deduction and planning has long been established for a variety of logics, e.g. temporal, linear and propositional logics, see [15, 6, 17, 12]. The existing work, however, focused on using effective theorem provers to build plans, rather than exploiting a planner to build a deduction. To the best of our knowledge, ours is the first experiment in this direction.

7 Conclusions and Future Work

In this paper, we have attempted to take seriously the idea that reasoning is a process which requires memory, and developed a framework for representing and verifying memory-bounded reasoners. 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 memory is explicitly modelled. The logic is interpreted on state transition systems, where the reasoner'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 memory required by the agent to derive a certain formula.

In future work, we plan to remove some idealisations made in the present work, such as constant size of formulas, and paying no penalty in terms of memory for backtracking.

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

References

1. T. Ågotnes and M. Walicki. Complete axiomatizations of finite syntactic epistemic states. In *Proceedings of the 3rd International Workshop on Declarative Agent Languages and Technologies (DALT 2005)*, July 2005.

2. 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.
3. N. Alechina, B. Logan, and M. Whitsey. A complete and decidable logic for resource-bounded agents. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2004)*, pages 606–613, New York, July 2004. ACM Press.
4. M. Benerecetti, F. Giunchiglia, and L. Serafini. Model Checking Multiagent Systems. *Journal of Logic and Computation, Special Issue on Computational & Logical Aspects of Multi-Agent Systems*, 8(3):401–423, 1998.
5. M. Berger, B. Bauer, and M. Watzke. A scalable agent infrastructure. In *Proceedings of the Second Workshop on Infrastructure for Agents, MAS and Scalable MAS (held in conjuction with Autonomous Agents'01)*, Montreal, 2001.
6. W. Bibel. A deductive solution for plan generation. *New Generation Computing*, 4:115–132, 1986.
7. 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.
8. H. N. Duc. Reasoning about rational, but not logically omniscient, agents. *Journal of Logic and Computation*, 7(5):633–648, 1997.
9. J. Elgot-Drapkin, M. Miller, and D. Perlis. Memory, reason and time: the Step-Logic approach. In *Philosophy and AI: Essays at the Interface*, pages 79–103. MIT Press, Cambridge, Mass., 1991.
10. J. J. Elgot-Drapkin and D. Perlis. Reasoning situated in time I: Basic concepts. *Journal of Experimental and Theoretical Artificial Intelligence*, 2:75–98, 1990.
11. R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge, Mass., 1995.
12. R. E. Fikes and N. J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:187–208, 1971.
13. M. Fisher and C. Ghidini. Programming Resource-Bounded Deliberative Agents. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 200–206. Morgan Kaufmann, 1999.
14. 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.
15. E. Jacopin. Classical AI planning as theorem proving: The case of a fragment of linear logic. In *AAAI Fall Symposium on Automated Deduction in Nonstandard Logics*, pages 62–66, Palo Alto, California, 1993. AAAI Press.
16. M. Kacprzak, A. Lomuscio, and W. Penczek. Verification of multiagent systems via unbounded model checking. In *Proceedings of the Third International Joint on Autonomous Agents and Multiagent systems. (AAMAS04)*, New York, August 2004.
17. H. Kautz and B. Selman. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1194–1201. AAAI Press, 1996.
18. S. J. Russell. Rationality and intelligence. *Artif. Intell.*, 94(1-2):57–77, 1997.
19. S. Tarkoma and M. Laukkanen. Supporting software agents on small devices. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02)*, pages 565–566, New York, NY, USA, 2002. ACM Press.
20. J. van Benthem and F. Liu. Diversity of agents in games. *Philosophia Scientiae*, 8(2), 2004.