Epistemic Reasoning in Jason

Michael Vezina
Carleton University
Ottawa, Ontario, Canada
michaeljvezina@cmail.carleton.ca

Babak Esfandiari
Carleton University
Ottawa, Ontario, Canada
babak@sce.carleton.ca

ABSTRACT

This paper presents an extension to the Jason BDI language to allow qualitative reasoning under uncertainty. We demonstrate the need for such an extension using a challenge from the 2019 Multi-Agent Programming Contest (MAPC), namely localization for navigation. Given the ability to qualitatively reason about what the agent knows and what it considers possible (or impossible), these challenges become easier to express, reason about, and act upon in a Jason program.

Through the use of epistemic logic and the epistemic reasoner in Hintikka's World, our extension allows agents to express epistemic queries; specifically, utilizing the class of single-agent S5 epistemic models to model-check queries about the agent's uncertainty. This paper also provides an evaluation of the overall performance and scalability of the extension's implementation to show how it impacts the agent's reasoning time; from the evaluation results, we use the official 2019 MAPC time constraints to examine the performance tradeoffs of using the presented extension to model and reason about uncertainty.

KEYWORDS

Epistemic reasoning; Jason; BDI; Epistemic Logic; Dynamic Epistemic Logic; Public Announcement Logic; Model-checking; Hintikka's World; Knowledge; Possibilities; AgentSpeak; Uncertainty; Model generation; Model updates; Jason extension

ACM Reference Format:
Michael Vezina and Babak Esfandiari. 2022. Epistemic Reasoning in Jason. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Online, May 9–13, 2022*, IFAAMAS, 9 pages.

1 INTRODUCTION

The belief-desire-intention (BDI) agent paradigm [12], and BDI-based languages such as Jason [5], provide practical approaches to defining autonomous agent behaviour by providing a reasoning cycle designed around the mental attitudes of an agent.

Agents must be developed such that they are resilient to uncertainty – one source of uncertainty, for example, occurs when the environment is partially-observable. Jason provides the ability to define contingency plans, which allow the agent to handle (or recover from) unexpected states or failures that result from uncertainty. However, the agent may prefer to avoid these unexpected states altogether; this is where the ability to reason about current uncertainty comes in handy. An agent's belief base allows it to reason about what it is explicitly certain (or uncertain) of, but does not natively infer other *possible* (or *impossible*) beliefs (which provides the ability to proactively handle uncertainty).

In this paper, we present a Jason extension that allows an agent to natively express knowledge and possibility (belief) queries by integrating the Jason architecture with a reasoner for *epistemic logic*[1]. We motivate the extension by showing how it could be used to reason about uncertainty in the *2019 Multi-Agent Programming Contest*, however, in general, our proposed extension can be used to develop a Jason program for any agent in knowledge-based domains where knowledge is modelled and reasoned about using *epistemic logic* and *public announcement events* – see [7].

1.1 The 2019 Multi-Agent Programming Contest

The Multi-Agent Programming Contest (MAPC) [2] is a yearly contest that brings together teams of agents to compete in a simulated environment. In the 2019 MAPC, agents must gather blocks, which can be requested via dispensers, to assemble various shapes specified by the simulation; communication and collaboration among agents of the same team are a necessity.

The agents are not given their absolute coordinates in the grid world but only their relative perception of their immediate environment. From this, various challenges arise that require reasoning about uncertainty such as *navigation* and *agent identification*; we have chosen to focus solely on navigation in this paper.

For demonstration purposes, we simplify the navigation challenge by assuming our agent starts with a complete definition of the map[3], as shown in Figure 1a – dark grey cells denote obstacles and the red square denotes a block dispenser. Locations on the map are referenced using (X, Y) coordinates, where (0, 0) is the top-left location and (4, 4) is the bottom-right. Figure 1b shows the agent's immediate perceptions, denoted by the green cells.

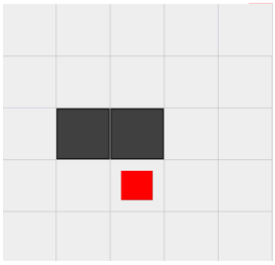
In this example, the agent wants to navigate to the red dispenser. We define the following Jason literals:

- *percept(Dir, Object)*: the agent knows it perceives *Object* (one of *obstacle* or *none*) in the direction *Dir* (one of *up, down, left, right*).
- *location(X, Y)*: the agent knows its current location on the map is (X, Y).
- *direction(Dir)*: the agent knows it must move in direction *Dir* to get to the red dispenser; this direction is computed using the shortest path.

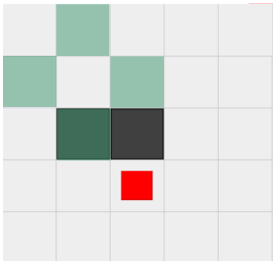
We would like to reason about which *location(X, Y)* and *direction(Dir)* values are possible given our perceptions. To do this, we

[1] We attempt to generalize our extension as much as possible, so that it could easily be adapted to other AgentSpeak or BDI languages.
[2] https://multiagentcontest.org/
[3] In the original MAPC, map definitions are randomly generated per simulation; as such, the agent must obtain the map definition via exploration.

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Online.



(a) The MAPC map definition. The dark grey cells are obstacles, and the red square denotes a block dispenser.

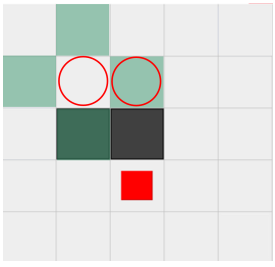


(b) The agent's perceived cells marked in green, given it's actual location of (1, 1).

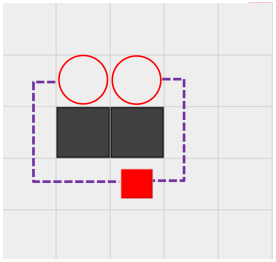
Figure 1: The MAPC map definition and agent perceptions.

want to express literal queries such as *possible(location(X, Y))* and *possible(direction(Dir))*, where *X*, *Y*, and *Dir* are unified with possible values. In Figure 2a and 2b, we have the following possibilities as a result of the agent's current perceptions, most notably because of *percept(down, obstacle)*:

- *possible(location(1, 1)), possible(location(2, 1))*
- *possible(direction(left)), possible(direction(right))*



(a) The agent's possible locations (red circles), given its perceptions.



(b) The shortest paths, marked with dashed lines, from each possible location to the red dispenser.

Figure 2: The MAPC map definition and agent perceptions.

Further, the agent would then like to express these possible queries in its plans; for example a *navigate* plan (Listing 1) that navigates the agent to its desired destination when it is not certain about its location or movement direction:

```
1 +!navigate
2     :   not direction(_) & possible(direction(Dir))
3     <-  move(Dir);
4         ...
```

Listing 1: A Jason plan that moves the agent in a possible direction.

This plan ensures that the agent moves in a direction that may possibly bring it to the red dispenser; although the direction may not guarantee a dispenser, the agent can be certain that it *does not* move in a direction that is known *not* to have a dispenser.

In essence, we want the agent to be able to query its possibilities, given what it becomes certain of. A naive approach would rely on the agent program itself, rather than the language, to derive and revise its possible beliefs based on the certain beliefs the agent holds. For example, when the agent's perceptions (i.e., what it is certain of) change, we revise all beliefs of the form *possible(direction(Dir))*.

Enter epistemic logic; epistemic logic is a multi-agent modal logic that can be used to query an agent's knowledge (certainty) or possibilities, via the model-checking of formulae. We propose an extension to Jason that allows us to reason about certainty and uncertainty of a single Jason agent, through the use of an epistemic logic reasoner. Despite the MAPC being a multi-agent contest, we only want to model and reason about the knowledge of a single agent.

2 SINGLE-AGENT EPISTEMIC LOGIC

Given a finite non-empty set of atomic propositions P, the syntax for a single-agent epistemic formula φ is defined as [1, 15]:

$$\varphi ::= p \mid \varphi \wedge \varphi \mid \neg\varphi \mid K\varphi$$
$$p \in P$$

From this, we also have disjunctions, i.e., $\varphi_1 \vee \varphi_2 \equiv \neg(\neg\varphi_1 \wedge \neg\varphi_2)$, and the possibility modality $K_{\text{Poss}}\varphi \equiv \neg K\neg\varphi$. $K\varphi$ and $K_{\text{Poss}}\varphi$ are read as "the agent knows φ" and "the agent considers φ to be possible", respectively.

Entailment semantics are provided using a *pointed epistemic model*: (M, w), where M is a single-agent *S5 epistemic model* and w is a *pointed world* chosen to provide correct entailment of formulae. $M = (W, R, V)$ [1, 15], where:

W is a finite set of worlds,

$R = W \times W$ is an indistinguishability relation for the agent,

$V : W \to 2^P$ is a valuation function

Given two worlds $(w_1, w_2) \in R$, we say that the agent is unable to distinguish world w_1 from w_2. R is modelled as an equivalence relation ($W \times W$) since all worlds are indistinguishable from each other, a consequence of being an S5 model. The valuation function V maps each world $w_v \in W$, to a subset of atomic propositions, such that $V(w_v) \subseteq P$; this represents the subset of P that hold true in the world w_v.

Given a pointed epistemic model (M, w) and an epistemic formula φ, entailment ($\models$) is defined as follows [1, 15]:

$(M, w) \models p$ iff $p \in V(w)$

$(M, w) \models \neg\varphi$ iff $(M, w) \not\models \varphi$

$(M, w) \models (\varphi_1 \wedge \varphi_2)$ iff $(M, w) \models \varphi_1 \wedge (M, w) \models \varphi_2$

$(M, w) \models K\varphi$ iff for all $(w, w') \in R$, $(M, w') \models \varphi$

Once a pointed model has been created, public announcement events can be used to update the model given new knowledge.

Public Announcement Events. Public announcements allow us to model updates to the agent's knowledge. Public announcements belong to the field of dynamic epistemic logic (DEL) [8], an extension of epistemic logic that provides the ability to model the impact of actions and events on agent knowledge.

Given a pointed epistemic model (M, w) and the public announcement $[\phi!]$ containing the agent's new knowledge represented by the epistemic formula ϕ, $[\phi!]$ must be executable ($(M, w) \models$

ϕ) in order to be applied. We apply $[\phi!]$ to (M, w), denoted by the $\otimes$ operator, which gives us a resultant pointed model (M_R, w):

$$(M_R, w) \leftarrow (M, w) \otimes [\phi!]$$

In the resultant model, only the worlds, indistinguishability relations, and proposition valuations from the original model, where the announced formula ϕ is true, are kept. The original pointed world, w, is kept as the pointed world for the resultant model. More formally, the resultant model $M_R = (W_R, R_R, V_R)$ is defined as follows [1]:

$$W_R = \{w \mid w \in W, \text{where } (M, w) \models \phi\}$$
$$R_R = R \cap (W_R \times W_R)$$
$$V_R(w) = V(w) \text{ for all } w \in W_R$$

3 EXTENDING JASON

Our extension provides a Jason agent with the ability to reason about its uncertainty through the use of epistemic logic and its corresponding K and K_{Poss} modalities. In order to allow the agent to query using these modalities, the extension must first create an epistemic model that describes the agent's domain of uncertainty. Also, as the agent executes its reasoning cycle, the knowledge it holds in its belief base is constantly being revised and updated; as such, the extension must update the epistemic model to ensure consistency with the belief base.

The extension aims to create, update, and query the epistemic model on behalf of the agent by assigning additional semantics to standard Jason syntax, this allows the extension to infer necessary information about the agent's domain of uncertainty while also minimizing the burden put on the developer to model and express queries about the agent's uncertainty.

We introduce the extension in the following sub-sections according to how it creates, updates, and queries the epistemic model; model creation occurs when a Jason agent is initialized (before its reasoning cycle starts), model updates are triggered whenever Jason's belief revision (BRF) or update (BUF) functions are executed[4], and model queries are triggered when the agent performs a belief query (e.g., in a plan context). Although we present the extension as one that integrates with Jason, it could easily be adapted to other AgentSpeak-based languages.

3.1 Model Creation

In order to infer an initial epistemic model from the agent's domain of uncertainty the agent can define its range of possibilities for a given literal using the extension's *range rules*, and model the relations between each range of possible values through positive and negative *valuation rules*. From the range and valuation rules defined by the agent, the extension performs various transformations that allow it to create an initial epistemic model.

Range Rules. The extension relies on the extended semantics it provides to range rules to define the "range" of possibilities that exist for a given literal. Let L^+ be a positive literal for which the agent wants to reason about its possibilities and let *Body* be a logical expression that grounds L^+ with all of its possible values: $\{l_1^+, \ldots, l_n^+\}$. Range rules are Jason rules that have the form:

$$\langle \text{Range Rule} \rangle \quad ::= \quad range(L^+) \text{ :- } Body.$$

Given the agent's set of range rules, R_{Range}, the function F_{Range} maps the range rule literal L_r^+ for all $r \in R_{Range}$ to its set of possible positive ground literals, e.g., $F_{Range}(L_r^+) \rightarrow \{l_{r_1}^+, \ldots, l_{r_n}^+\}$.

In Listing 2, we show two range rules that define the set of possible locations and perceptions using Jason's built-in internal action *.member*[5], the range literals are mapped to its ground values as: F_{Range}(*location(X, Y)*) $\rightarrow$ {*location(0, 0), location(0, 1)*} and F_{Range}(*percept(Dir,Obj)*) $\rightarrow$ {*percept(up,none), percept(up,obstacle)*}.

```
1 range(location(X,Y))
2     :- .member([X,Y], [[0,0], [0,1]]).
3 range(percept(Dir,Obj))
4     :- .member([Dir,Obj], [[up,none], [up,obstacle]]).
```

Listing 2: Two example range rules.

From these range rules, the extension can infer an initial set of positive ground literal valuations called V_L — this is a set of literal valuations that will be transformed into an initial epistemic model; in the interim, we use literals so that we can apply the necessary transformations for valuation rules. The function that creates the initial V_L from the agent's range rules is shown in Algorithm 1; given any function F, we obtain F's range as *ran(F)*.

Function $V_L(F_{Range}, R_{Range})$
  $L_1^+, \ldots, L_{|R_{Range}|}^+ \leftarrow ran(F_{Range})$;
  return $L_1^+ \times \cdots \times L_{|R_{Range}|}^+$;
end

Algorithm 1: Infer the initial valuations V_L using the agent's R_{Range} and F_{Range}.

The initial V_L set inferred from the range rules allows us to create a generic epistemic model where a world is created containing one possible value from each range rule. This inferred set may not appropriately reflect the agent's domain of uncertainty (for example, the range rules in Listing 2 generate extra worlds where locations do not match their appropriate perceptions). As such, the extension assigns additional semantics to Jason rule syntax to append to (or restrict) these worlds, these are known as *positive and negative valuation rules*.

Positive and negative valuation rules are Jason rules that define the knowledge relationships between the possible values defined by each of the range rules; the valuation rules are interpreted by the extension and are reflected by the literals held by each valuation in V_L. Let L^+ be a positive literal that unifies with one (or more) of the agent's defined range literals, and let *Body* be a logical expression containing literals defined by any of the range rules:

$$\langle \text{Positive Valuation Rule} \rangle \quad ::= \quad L^+ \text{ :- } Body.$$
$$\langle \text{Negative Valuation Rule} \rangle \quad ::= \quad \sim L^+ \text{ :- } Body.$$

Listing 3 shows an example of a positive and negative valuation rule, respectively.

[4]To clarify, the extension uses the BRF/BUF to update the epistemic model such that it reflects the corresponding revision or update. The proposed extension **does not** perform automatic belief revision, the Jason program is still responsible for maintaining the consistency of the agent's current beliefs.

[5]E.g., *.member(X, List)* unifies 'X' with an element in the list 'List'. See http://jason.sourceforge.net/api/jason/stdlib/member.html.

```
percept(up,none) :- location(0,0).
~percept(up,obstacle) :- location(0,1).
```

Listing 3: An example of two valuation rules.

Positive Valuation Rules. Positive valuation rules are used to denote which literals belong in a given valuation. For each literal valuation in V_L generated using the range rules, we modify its literals based on the positive valuation rules defined by the agent.

Given a valuation $v \in V_L$ containing positive ground literals and a positive valuation rule R_{PVR}, the function $F_{PVR}(v, R_{PVR})$ returns a transformed valuation v' which replaces v in V_L. The ground literals defined by R_{PVR} will remove any existing literals in v that were inferred from the same range rule. Additionally, the function *consequences*(v, R_{PVR}) returns R_{PVR}'s ground head literals, obtained by evaluating the logical consequences of R_{PVR} with v.

The positive valuation rule shown in Listing 3 denotes that valuations with *location(0,0)* should always hold *percept(up,none)*, replacing any other values from the same range rule: *percept(Dir, Object)*.

Negative Valuation Rules. Once the positive valuation rules have processed the valuations in V_L, we apply any negative valuation rules. For each literal valuation in V_L, we modify its literals based on the negative valuation rules defined by the agent. Given a valuation $v \in V_L$ and a negative valuation rule R_{NVR}, the function $F_{NVR}(v, R_{NVR})$ returns a transformed valuation v' which replaces v in V_L; if $v' == \emptyset$, then the valuation is removed from V_L altogether. This removal occurs when the valuation does not contain at least one element from each of the range rule's possible values.

The negative valuation rule shown in Listing 3 denotes that valuations with *location(0,1)* should not hold *percept(up,obstacle)*, removing the world if it does not hold any other values from the same range rule: *percept(Dir, Object)*. The functions F_{PVR} and F_{NVR} are shown in Algorithm 2.

Given the transformed set of literal valuations V_L, Algorithm 3 creates an initial epistemic model (M_I, w_I) using the process of *propositionalization*.

Propositionalization. We use a propositional epistemic logic for complexity purposes; as the reader will see in a later section, we use the epistemic reasoner in *Hintikka's World* [14] which provides P-Complete model-checking for epistemic formulae and public announcements [15]. Jason, on the other hand, uses ground literals for its beliefs. Since we want to model and reason about beliefs using epistemic logic, the extension must provide a way to transform ground literals into propositions [13].

Given a ground literal l, $prop(l)$ creates a propositional symbol for the non-negated form of the literal (l^+), e.g., $prop(location(0, 1))$ $\rightarrow$ location[0, 1] (we replace parentheses with square brackets to denote the propositionalized form of a literal).

After processing the transformations made by the range and valuation rules shown in Listings 2 and 3, the set V_L will contain two valuations; these are transformed into the epistemic model shown in Figure 3 where the worlds (locations) and their corresponding proposition valuation are shown. The indistinguishability relation is not shown in the figure as it is implicit (i.e., an equivalence relation among the worlds).

Function $F_{PVR}(v, R_{PVR})$
  $v' \leftarrow v$;
  $L \leftarrow consequences(v, R_{PVR})$;
  for $l \in L$ **do**
    for $L_R \in ran(F_{Range})$ *where* $l \in L_R$ **do**
      $v' \leftarrow v' \setminus L_R$;
    end
  end
  return $v' \cup L$;
end
Function $F_{NVR}(v, R_{NVR})$
  $v' \leftarrow v$;
  $L \leftarrow consequences(v, R_{NVR})$;
  for $l \in L$ **do**
    $v' \leftarrow v' \setminus \{l\}$
    for $L_R \in ran(F_{Range})$ *where* $l \in L_R$ **do**
      if $v' \cap L_R = \emptyset$ **then**
        return $\emptyset$
      end
    end
  end
  return v'
end

Algorithm 2: The functions F_{PVR} and F_{NVR} for applying a positive (R_{PVR}) or negative (R_{NVR}) valuation rule, respectively, to a given literal valuation $v \in V_L$.

Function *createModel*(V_L)
  $W \leftarrow \emptyset$
  for $v_L \in V_L$ **do**
    $w \leftarrow$ New World
    $W \leftarrow W \cup \{w\}$
    $V(w) \leftarrow \{prop(l^+) \mid l^+ \in v_L\}$
  end
  $R \leftarrow W \times W$
  $M_I \leftarrow (W, R, V)$
  $w_I \leftarrow$ choose any $w \in W$
end

Algorithm 3: Create the initial model (M_I, w_I) from the set V_L.

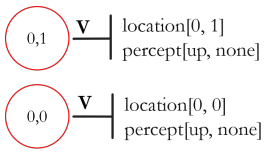


Figure 3: The created initial epistemic model.

3.2 Model Updates

When the agent updates the knowledge it holds in its belief base, whether from a belief revision or update, the extension must synchronize the epistemic model by applying a public announcement that reflects the corresponding update.

When an update occurs, we use the sets BB_{Prev} and BB_{Cur} to represent the set of beliefs held by the belief base before the update and after the update, respectively.

Public announcements are monotonic knowledge updates, as they append to the knowledge held by the epistemic model they are applied to. That is, public announcements model an update in knowledge that maintains the agent's previous knowledge ($BB_{Prev} \subseteq BB_{Cur}$), since they are typically applied to the current epistemic model (M_C, w_c). However, when a non-monotonic update occurs ($BB_{Prev} \not\subseteq BB_{Cur}$), we can apply it in a pseudo-monotonic manner by executing the announcement on the initial epistemic model (M_I, w_I), which, unlike the current model, does not model previous knowledge or announcements. Algorithm 4 selects the appropriate model (M_A) based on the type of knowledge update.

if $BB_{Prev} \subseteq BB_{Cur}$ **then**
  $M_A \leftarrow M_C$
else
  $M_A \leftarrow M_I$
end

Algorithm 4: Choose the correct model for the public announcement.

We then create a conjunction of propositions representing the agent's certainty in BB_{Cur}, choose a new pointed world w_A where the knowledge holds true, and apply this formula as a public announcement to the model; this is shown in Algorithm 5.

Function *createFormula*(BB_{Cur})
  $BB_{\text{Prop}} \leftarrow \emptyset$
  for $b \in BB_{Cur}$ **do**
    if *b is negated* **then**
      $BB_{\text{Prop}} \leftarrow BB_{\text{Prop}} \cup \neg prop(b)$
    else
      $BB_{\text{Prop}} \leftarrow BB_{\text{Prop}} \cup prop(b)$
    end
  end
  return $\bigwedge_{b \in BB_{\text{Prop}}}$
end
Function *updateModel*(BB_{Cur}, M_A)
  $\phi \leftarrow createFormula(BB_{Cur})$
  $W_A \leftarrow$ set of worlds in M_A
  $w_A \leftarrow w \in W_A$ where $(M_A, w) \models \phi$
  $(M_C, w_C) \leftarrow (M_A, w_A) \otimes [\phi!]$
end

Algorithm 5: The functions *createFormula* and *updateModel*, which create an update formula and update the epistemic model, given the agent's set of beliefs BB_{Cur}.

3.3 Model Queries

Given a positive Jason literal L^+ that unifies with one or more ground range literals, we define epistemic query literals using the following BNF:

$$\begin{aligned} \langle \text{Query} \rangle &::= \sim L^+ \mid L^+ \\ \langle \text{KnowQuery} \rangle &::= \langle \text{Query} \rangle \\ \langle \text{PossQuery} \rangle &::= possible(\ \langle \text{Query} \rangle\ ) \\ \langle \text{EpistemicQuery} \rangle &::= \langle \text{KnowQuery} \rangle \mid \langle \text{PossQuery} \rangle \end{aligned}$$

All epistemic queries are transformed so that they can be evaluated as epistemic formulae; non-epistemic belief queries are forwarded to Jason's default belief base. Given an epistemic belief query Q (and its underlying literal L^+), the set Q_G contains all groundings of Q, where its L^+ is unified using ground range literals. For each ground literal $q_G \in Q_G$, where l is the value that grounded L^+, we transform q_G into a corresponding epistemic formula ϕ according to the transformations shown in Table 1.

Table 1: Transforming a ground epistemic query q_G, and its ground range literal l into an epistemic formula ϕ.

q_G	ϕ
l	$K\ prop(l)$
$\sim l$	$K\ \neg prop(l)$
$possible(l)$	$K_{\text{Poss}}\ prop(l)$
$possible(\sim l)$	$K_{\text{Poss}}\ \neg prop(l)$

4 CONNECTING TO HINTIKKA'S WORLD

Hintikka's World [14, 15] was developed by François Schwarzentruber as a pedagogical tool for DEL. We use Hintikka's World with our extension as it provides an open-source, easy-to-use implementation for modelling and reasoning about DEL. In this section, we show how we can create, update, and query the epistemic model using Hintikka's World.

4.1 Model Creation

Given the initial pointed epistemic model's components (W, R, V, w_I) inferred by the extension, we can create the initial model (M_I, w_I) in Hintikka's World. We use the *ExplicitEpistemicModel* type provided by Hintikka's World to represent the agent's epistemic model. Algorithm 6 shows the creation of a model instance $E_{Initial}$, which represents (M_I, w_I). In Hintikka's World, we also use the instance $E_{Current}$ to refer to the current model (M_C, w_C), and we use the constant *AG* to refer to our agent.

4.2 Model Updates

When the agent updates the knowledge it holds in its belief base, we construct and apply a public announcement $[\phi!]$ to a chosen epistemic model (M_A, w_A), i.e., $(M_A, w_A) \otimes [\phi!]$.

Hintikka's World provides a *Formula* type and various sub-types (*AndFormula*, *KFormula*, etc.) to represent an epistemic formula ϕ. We define a function *toHWFormula*: $\phi \rightarrow$ *Formula*, which constructs an instance of *Formula* that represents the epistemic formula ϕ.

Algorithm 7 demonstrates how the extension performs the model update $(M_A, w_A) \otimes [\phi!]$ in Hintikka's World. We use E_A as the Hintikka's World representation for (M_A, w_A).

Function *createModel*(W, R, V, w_I)
  $E_{Initial} \leftarrow$ new *ExplicitEpistemicModel*();
  for $w \in W$ **do**
    $E_{Initial}$.addNode(w, new *ValuationWorld*($V(w)$));
  end
  for $(w_1, w_2) \in R$ **do**
    $E_{Initial}$.addEdge(AG, w_1, w_2);
  end
  $E_{Initial}$.setPointedNode(w_I);
  $E_{Current} \leftarrow E_{Initial}$;
end

Algorithm 6: Create the initial epistemic model in Hintikka's World, using the generated components (W, R, V) and w_I.

Function *updateModel*(E_A, ϕ)
  $\phi_{Ann} \leftarrow$ *toHWFormula*(ϕ)
  $[\phi_{Ann}!] \leftarrow$ ExplicitEventModel.
   getEventModelPublicAnnouncement(ϕ_{Ann}, [AG])
  $E_{\text{Current}} \leftarrow [\phi_{Ann}!]$.apply($E_A$)
end

Algorithm 7: Apply an announcement $[\phi_{Ann}!]$ to the chosen model (E_A).

4.3 Model Querying

When a Jason agent sends a knowledge query to the epistemic reasoner via an API request, we use the current epistemic model to evaluate the query (using model checking provided by Hintikka's World) and send the evaluation results back to the agent. Model-checking a formula ϕ_q given the current model $E_{Current}$ is shown in Algorithm 8.

Function *queryModel*$(E_{Current}, \phi_q)$
  $\phi \leftarrow$ *toHWFormula*(ϕ_q)
  return $E_{Current}$.*modelCheck*(ϕ)
end

Algorithm 8: Model-check a formula ϕ_q using the current epistemic model $E_{Current}$.

In the next section we apply the framework to the MAPC navigation challenge we introduced earlier.

5 APPLICATION: MAPC NAVIGATION

We aim to use our proposed Jason extension to reason about the locations that the agent currently considers possible, based on its current knowledge (perceptions). We assume that the agent is given the map definition beforehand. Figure 4 shows the map definition given to the agent; since the agent has not received any perceptions, all locations are currently possible and are denoted on the figure with red circles.

Model Creation. Given the MAPC map, we can use standard Jason rules with our extended semantics to define the agent's initial domain of uncertainty; Listing 4 shows the range rules that define all possible locations and perceptions, and the valuation rules that link perceptions and directions to their appropriate locations.

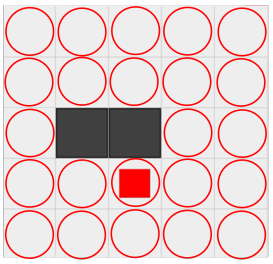


Figure 4: The agent's initial possible locations.

```
1  // Range Rules
2  range(location(X, Y))
3      :- .member([X,Y], [[0,0], ..., [4,4]]).
4  range(percept(Dir, Obj))
5      :- .member([Dir,Obj], [[up,none], ...]).
6  range(direction(Dir))
7      :- .member(Dir, [up, down, left, right]).
8
9  // Valuation Rules
10 percept(Dir,Obj) :- location(1, 1) & .member([Dir,Obj],
       [[up,none], [down,obstacle], [left,none],
       [right,none]]).
11 ...
12
13 direction(left) :- location(1, 1).
14 ...
```

Listing 4: MAPC range and valuation rules.

The extension then uses these range and valuation rules to infer the initial epistemic model shown in Figure 5; due to the rules specified by this agent, each world corresponds to a location on the map. The figure shows the worlds that are generated, and the valuation for the world with location (1,1) is shown in blue; the relation R is implicit and is thus not shown.

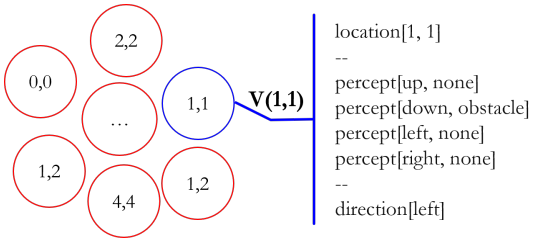


Figure 5: Generated initial model.

Belief Update. Now assuming the MAPC simulation starts and the agent is placed at location (1, 1) on the map, it receives its appropriate perceptions via Jason's *perceive* function: *percept(up, none), percept(down, obstacle), percept(right, none), percept(left, none).* This situation was shown previously in Figure 2a. These new perceptions are propositionalized and are used to create the public announcement: $[percept[up, none] \wedge percept[down, object] \wedge \ldots!]$ This eliminates any worlds where these perceptions do not hold true; as such, the resulting worlds are shown in Figure 6.

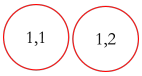


Figure 6: The updated model.

Belief Queries. We will now demonstrate the evaluation of the query *possible(direction(Dir))* from the plan context shown in Listing 1. This query is ground using the agent's ground range literals, which are transformed into their corresponding epistemic formulae:

- *possible(direction(left))* $\longrightarrow K_{\text{Poss}}$ *direction[left]*
- *possible(direction(right))* $\longrightarrow K_{\text{Poss}}$ *direction[right]*
- *possible(direction(down))* $\longrightarrow K_{\text{Poss}}$ *direction[down]*
- *possible(direction(up))* $\longrightarrow K_{\text{Poss}}$ *direction[up]*

Assuming the agent's current model (M_C, w_C) is the one shown in Figure 6, only the following are true: $(M_C, w_C) \models K_{\text{Poss}}$ *direction[left]* and $(M_C, w_C) \models K_{\text{Poss}}$ *direction[right]*. The reader may refer to Figure 2b to see why this is the case. As such, the ground literals *possible(direction(left))* and *possible(direction(right))* will be used to ground the agent's original query *possible(direction(Dir))*; note that the process for a knowledge query, e.g., *direction(_)*, is the exact same, but creates formulae with the modality K.

6 EVALUATION

In this section, we evaluate the performance and scalability of the extension to measure its impact on the agent's reasoning cycle[6].

6.1 Model Creation

During model creation, the extension creates an initial model using the transformations defined by our assigned range and valuation rule. We measure the impact the size of the created epistemic model has on model creation time, represented by the model parameters $|W|$ (the number of worlds) and $|V_W|$ (the number of propositions in each world).

We expect the major time-complexity factors to be driven by the program-dependent evaluation of logical consequences for range and valuation rules (F_{Range} in Algorithm 1 and the *consequences()* function used in Algorithm 2) and the quadratic complexity associated with creating the indistinguishability relation (Algorithm 3).

In our MAPC application, $|W|$ depends on the size of the map; we therefore create models using various map sizes ranging from 5x5 ($|W| = 25$) to 100x100 ($|W| = 10,000$). The value of $|V_W|$ represents the number of perceptions per location, which we vary using extreme values of 4 and 100. Figure 7 shows the impact these parameters have on the model creation time.

6.2 Model Updates

Given the agent's current belief base BB, the extension performs model updates by creating a public announcement formula from BB and applying it to either the initial or current model (represented by M_A). To apply the update, the reasoner must evaluate the announcement containing $|BB|$ propositions on the worlds in the model: W_A.

[6]The results reported in this section were obtained with an Intel i7-8700K CPU and 48GB DDR4 RAM, running Windows 10.

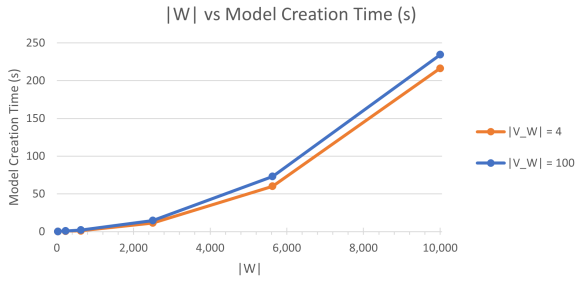


Figure 7: $|W|$ and $|V_W|$ vs. Model Creation Time (s). The orange and blue lines show models with $|V_W| = 4$ and $|V_W| = 100$, respectively.

We vary the parameter $|W_A|$ using the same values as the model creation evaluation (i.e., values of 25 to 100) and vary the belief base size using arbitrary values of $|BB| = 4$ and 4000. We expect model updates to have a quadratic time complexity, due to the need to iterate over the quadratic indistinguishability relation in the model M_A. Fortunately, we can improve this complexity with an optimized epistemic model implementation for the class of models we are using.

6.2.1 Optimized Epistemic Models. Creating and updating the epistemic model using Hintikka's World currently operates in quadratic time with the number of worlds in the model ($|W|^2$); this is due to having to iterate over the equivalence indistinguishability relation $R = W \times W$. However, we can modify Hintikka's World to infer this relation rather than create and store it explicitly.

In Hintikka's World, we use the *ExplicitEpistemicModel* type to represent the epistemic model using a graph structure (nodes are worlds/valuations and the edges are the indistinguishability relation); this is because multi-agent DEL allows for different classes of epistemic models and events. However, by modifying the implementation of *ExplicitEpistemicModel* to only support single-agent S5 models, we are able to obtain linear model update times [16]. Unfortunately, we do not receive any improvement in model creation times as it is primarily driven by the transformations made by the range and valuation rules rather than the initialization of the model. The original (quadratic) and optimized (linear) model update times are shown in Figure 8.

6.3 Model Queries

Lastly, the extension performs model queries by transforming a literal query into one or more propositional epistemic formulae and evaluating each of them using the current model (M_C, w_C). We measure the impact of a single query literal on the agent's reasoning cycle time, where $|W|$ is the number of worlds in the current model M_C (varied from 25 to 10,000), and $|F|$ is the number of formulae that get evaluated (where we use arbitrary values of $|F| = 5$ or 50). We define $|E| = |W| \times |F|$ as the number of evaluations that get made on the model to return the query results back to the agent. Figure 9 shows the impact of the model size and number of formulae on the total model querying time.

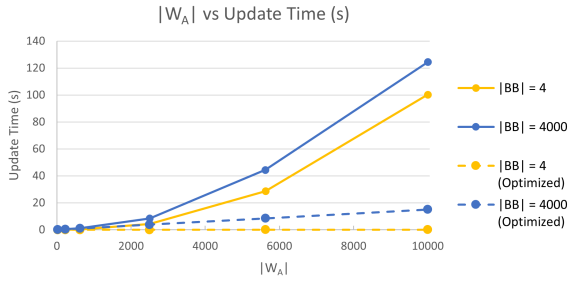


Figure 8: $|W_A|$ vs. Model Update Time (s), where the solid and dashed lines represent the original and optimized implementation, and the yellow and blue lines represent updates where $|BB| = 4$ and $|BB| = 4000$, respectively.

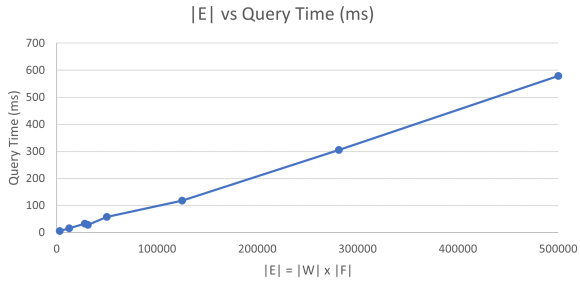


Figure 9: $|E|$ vs. Model Query Time (ms).

In the official MAPC simulation, the agents operate on a 50x50 map ($|W| = 2500$), and have 61 perceptions per location ($|V_W| = 61$). This gives them a model creation, update, and query time of 14s, 75ms, and 16 ms ($|F| = 4$), respectively. Each simulation step has a 4 second deadline to choose an action. Since model creation takes 14s, the agent may miss the first 3 simulation steps; however, since this is a one-time process and model updates and queries execute well within the deadline, the agent will make up for lost simulation steps since it gains the ability to proactively reason about (and act on) its uncertainty.

7 RELATED WORK

In the literature, there have been attempts to integrate epistemic logic with other AI approaches, e.g., the epistemic rule-based approach presented in [9], and epistemic planning for multi-agent systems [4], however, to our knowledge, there have been no attempts to extend an agent-oriented programming language such as AgentSpeak with an epistemic reasoner.

In [6], a theoretical BDI extension named g-BDI is proposed. The agent's belief, desire, and intention modalities are graded, and can be used to qualitatively model and reason about its uncertainty. Inspired by g-BDI, a practical (i.e., implemented) Jason extension named G-JASON [3] was developed, however, G-JASON takes a different approach to modelling uncertainty by associating beliefs, desires, and intentions with a numerical value of uncertainty rather than g-BDI's graded modalities.

Another practical approach to uncertainty in the literature, named TEAgentSpeak [2], is an extension of AgentSpeak that provides the ability to model and reason about belief uncertainty through plausibility models. Reasoning about certainty or possibility of a belief is expressed via numerical values, rather than the modalities we presented here.

Although numerical methods (such as those used by G-JASON and TEAgentSpeak) may be more precise than a modal approach (e.g., via epistemic logic) to uncertainty, numerical approaches may not always be suitable for a given application – such as when the agent wants to qualitatively reason about the domain, or when the numerical values required to model the domain's uncertainty are simply not obtainable.

8 CONCLUSION: LIMITATIONS AND FUTURE WORK

This paper presented an extension to the Jason language, which enabled the ability to epistemically reason about an agent's current beliefs using the K and K_{Poss} modalities. The extension's assigned semantics allow it to infer the agent's domain of uncertainty via range and valuation rules, and from them, create an initial epistemic model. As the agent executes its reasoning cycle, the extension will update and query the epistemic model based on the belief updates and queries performed by the agent. Using the MAPC navigation challenge, we demonstrated how such an extension could be used to proactively reason about uncertainty.

When using our extension, the agent may find itself in situations where it wishes it could express other domain-specific epistemic model transformations. For example, in the official MAPC simulation, the agent is not given a map definition ahead of time and is expected to explore the map; however, our extension does not provide any way to append new worlds and valuations to the model to accommodate discovered locations and perceptions. This means that the developer must anticipate and model all possible situations before the initial model is created.

Additionally, when a non-monotonic update occurs (e.g., receiving new perceptions as a result of moving locations), the model may contain less certainty about the new situation as a result of how non-monotonic updates are applied. Currently, the agent must use an ad hoc approach after the update is applied to transition its previous level of certainty to a new situation, however, as future work, we would like the extension to infer these transitions via additional extended syntax and/or semantics.

To address the aforementioned limitations, our extension needs to support additional epistemic model transformations that allow the agent to express other types of model updates; luckily, this is why we have chosen to integrate with Hintikka's World, a reasoner that supports the full DEL language. DEL has the ability to model the impact of various types of actions and events on the agent's knowledge, however, supporting these additional transformations will come at the cost of additional computational complexity [15]. Future work thus includes addressing these limitations while measuring the impact that each type of transformation has on the agent's reasoning cycle.

Lastly, as future work, our epistemic extension could be enriched with the ability to perform automated belief revision, for example, using the approaches presented in [10, 11].

REFERENCES

[1] Alexandru Baltag and Bryan Renne. 2016. Dynamic Epistemic Logic. In *The Stanford Encyclopedia of Philosophy* (winter 2016 ed.), Edward N. Zalta (Ed.). Metaphysics Research Lab, Stanford University.

[2] Kim Bauters, Kevin McAreavey, Weiru Liu, Jun Hong, Lluis Godo, and Carles Sierra. 2017. Managing Different Sources of Uncertainty in a BDI Framework in a Principled Way with Tractable Fragments. *Journal of Artificial Intelligence Research* 58 (04 2017), 731–775. https://doi.org/10.1613/jair.5287

[3] Adrián Biga and Ana Casali. 2014. *G-JASON: An Extension of JASON to Engineer Agents Capable to Reason under Uncertainty.* 17–28.

[4] Thomas Bolander. 2017. A Gentle Introduction to Epistemic Planning: The DEL Approach. *Electronic Proceedings in Theoretical Computer Science* 243 (Mar 2017), 1–22. https://doi.org/10.4204/eptcs.243.1

[5] Rafael Bordini, Jomi Hübner, and Michael Wooldridge. 2007. *Programming Multi-Agent Systems in AgentSpeak Using Jason.* Vol. 8. https://doi.org/10.1002/9780470061848

[6] Ana Casali, Lluís Godo, and Carles Sierra. 2009. g-BDI: A Graded Intensional Agent Model for Practical Reasoning. In *Modeling Decisions for Artificial Intelligence*, Vicenç Torra, Yasuo Narukawa, and Masahiro Inuiguchi (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 5–20.

[7] Ronald Fagin, Joseph Halpern, Yoram Moses, and Moshe Vardi. 2003. *Reasoning About Knowledge.* https://doi.org/10.7551/mitpress/5803.001.0001

[8] Barteld Kooi Hans van Ditmarsch, Wiebe van der Hoek. 2008. *Dynamic Epistemic Logic.* Springer. http://gen.lib.rus.ec/book/index.php?md5=20f4c68a27ae7d5ab483511401c9532f

[9] Mark Jago. 2009. Epistemic Logic for Rule-Based Agents. *Journal of Logic, Language and Information* 18, 1 (01 Jan 2009), 131–158. https://doi.org/10.1007/s10849-008-9071-8

[10] Emiliano Lorini. 2020. Rethinking epistemic logic with belief bases. *Artificial Intelligence* 282 (2020), 103233. https://doi.org/10.1016/j.artint.2020.103233

[11] Emiliano Lorini and Francois Schwarzentruber. 2021. Multi-Agent Belief Base Revision. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, Zhi-Hua Zhou (Ed.). International Joint Conferences on Artificial Intelligence Organization, 1959–1965. https://doi.org/10.24963/ijcai.2021/270 Main Track.

[12] Anand Rao and Michael Georgeff. 2000. BDI Agents: From Theory to Practice. (11 2000).

[13] Stuart Russell and Peter Norvig. 2009. *Artificial Intelligence: A Modern Approach* (3rd ed.). Prentice Hall Press, USA.

[14] François Schwarzentruber. 2018. Hintikka's World: Agents with Higher-order Knowledge. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18.* International Joint Conferences on Artificial Intelligence Organization, 5859–5861. https://doi.org/10.24963/ijcai.2018/862

[15] Francois Schwarzentruber. 2019. *Epistemic reasoning in Artificial Intelligence.* Habilitation Thesis.

[16] Michael Vezina. 2022. Epistemic Reasoning Module for Jason (Source Code). (01 2022). https://doi.org/10.5281/zenodo.5851517