ALGORITHM FOR MODAL-CLAUSE RESOLUTION TO SUPPORT EPISTEMIC REASONING IN MULTI-AGENT SYSTEMS

An Undergraduate Research Scholars Thesis

by

BRYSON KYLE MROSKO

Submitted to the LAUNCH: Undergraduate Research office at
Texas A&M University
in partial fulfillment of the requirements for the designation as an

UNDERGRADUATE RESEARCH SCHOLAR

Approved by
Faculty Research Advisor: Dr. Thomas Ioerger

May 2023

Majors: Computer Science
Applied Mathematics - Cryptography

RESEARCH COMPLIANCE CERTIFICATION

Research activities involving the use of human subjects, vertebrate animals, and/or biohazards must be reviewed and approved by the appropriate Texas A&M University regulatory research committee (i.e., IRB, IACUC, IBC) before the activity can commence. This requirement applies to activities conducted at Texas A&M and to activities conducted at non-Texas A&M facilities or institutions. In both cases, students are responsible for working with the relevant Texas A&M research compliance program to ensure and document that all Texas A&M compliance obligations are met before the study begins.

I, Bryson Kyle Mrosko, certify that all research compliance requirements related to this Undergraduate Research Scholars thesis have been addressed with my Faculty Research Advisor prior to the collection of any data used in this final thesis submission.

This project did not require approval from the Texas A&M University Research Compliance & Biosafety office.

TABLE OF CONTENTS

Page

ABSTRACT

Algorithm for Modal-Clause Resolution to Support Epistemic Reasoning in Multi-Agent Systems

Bryson Kyle Mrosko
Department of Computer Science & Engineering
Texas A&M University

Faculty Research Advisor: Dr. Thomas Ioerger
Department of Computer Science & Engineering
Texas A&M University

The ability to reason about other agents' beliefs has many applications in multi-agent systems, including facilitating coordinated behavior, teamwork, and information exchange. Modal logics are commonly used to formally capture and reason about knowledge/belief states. Unlike in propositional logic, it is computationally complex to create an automatic deduction algorithm capable of proving entailment when modal operators are considered. In this paper, we present an extension of a propositional logic resolution theorem prover to resolve modal clauses based on a complete set of inference rules for KD that was previously published by Enjalbert and del Cerro in 1989. We demonstrate that this modal resolution theorem prover, EpiRes, can be used for epistemic belief reasoning in an Information Privacy application domain by showing what an observer can infer by monitoring the actions of another agent. In this setting, a robot inspects nuclear reactor facilities for high or low radiation levels in varying locations depending on the reactor type. The goal of the observer is to try to infer the reactor type by monitoring the actions of the robot, which requires reasoning about the robot's goals, information obtained by actions, and justifying why the

robot took the actions it took. To more easily model this scenario, we created a modal, meta operator "knows whether" (KW) for writing rules that are agnostic to the true state of the world and provide the equivalent translation in KD. We conclude by showing that EpiRes was successful in showing what information can or cannot be inferred by the observer from different actions (plans) of the robot, without having to model belief states explicitly as sets of possible worlds.

DEDICATION

To my family, instructors, and peers who supported me throughout the research process.

ACKNOWLEDGMENTS

Contributors

I would like to thank my faculty advisor, Dr. Thomas Ioerger, and Dr. Dylan Shell, an additional contributor and reviewer, for their guidance and support throughout the course of this research.

Thanks also go to my friends and colleagues and the department faculty and staff for making my time at Texas A&M University a great experience.

EpiRes, the resolution-refutation-based theorem-proving system for modal-propositional knowledge bases, was developed jointly between Dr. Thomas Ioerger and myself. Dr. Dylan Shell donated his time in reviewing the work we had done and verifying it's faithfulness to implementing the foundational rules EpiRes relies on.

All other work conducted for the thesis was completed by the student independently.

Funding Sources

This research did not receive funding; Its contents are solely the responsibility of the authors and do not necessarily represent the official views of any other parties.

NOMENCLATURE

MC	Muddy Children
CNF	Conjunctive normal form
SNF	Separated normal form
EpiRes	Epistemic Reasoner
AST	Abstract syntax tree
KW	Knows whether

1. INTRODUCTION

1.1 Modal Logic

Multi-agent systems are a prime candidate for displaying the utility of reasoning in modal logics [Fagin *et al.*1995, Cohen and Levesque1990, Wooldridge and Lomuscio2000, Kominis and Geffner2015]. Consider scenarios in which agents must coordinate with each other, exchange information, or even work against each other. Any valuable ally, or enemy, considers information such as the goals, intentions, motivations, and beliefs of others. When considering non-observable information about others, it is only natural that these conclusions exist with a degree of uncertainty. Agents may know something to be a fact or agents may know something to be possible given other conditions.

Modal logics, in particular, are especially apt in expressing such statements which are unable to be represented in propositional logic. The representation of knowledge and beliefs in (multi-)modal logics is often represented with operators such as $\Box_i$ which can by applied to subsentences ϕ in a language where $\Box_i(\phi)$ (or $K_i(\phi)$), represents that agent i knows ϕ. In particular, the logic S5 is often used to model knowledge, and KD is often taken to be a reasonable approximation for representing beliefs of agents [Hintikka1962].

1.2 Automated Deduction Systems

There are several kinds of automated deduction systems that can be used to show the entailment of a query. Algorithms such as forward-chaining and backward-chaining use intuitive inference steps to move from premise to implications and vice versa. In forward-chaining, given facts are identified and all valid conclusions are drawn and added to the lists of facts which continues repeatedly until the query is added or no further inferences can be made. Backwards-chaining works similarly, however it considers all possible scenarios in which the query is entailed and considers the premises required to make such an inference repeating the process until it reaches a set of known facts that form the basis of the chain of reasoning.

In this paper, we focus on the other main type of automated deduction algorithms called resolution refutation. By applying a set of transformations to a propositional knowledge base, the sound inference rule of resolution can be applied repeatedly until it can be shown that a query is entailed. Each resolution step resolves two clauses together containing opposing literals, cancels them, and collects the remaining literals into a new, single clause containing one less literal. To show that a query is entailed, the negated query is asserted and the goal is to reach the empty clause by repeatedly resolving clauses together canceling literals until there is nothing left. By showing a contradiction is reached when the negated query is asserted, then it must be the case that the query is entailed.

1.3 Reasoning in a Modal Logic

In general, there are several methods available to automatically reason in modal logics, however they typically have high computational complexity [Halpern and Moses1992]. Such examples include model-checking [van Benthem *et al.*2018], tableau systems [Gasquet *et al.*2005], BDI [Rao1996], and translations into boolean satisfiability [Sebastiani and Vescovi2009]. Particularly, resolution refutation theorem provers have been used to reason within modal logics, however they typically require complex transformations. Some approaches consider translations into separated normal form (SNF) [Dixon *et al.*1998] while others use a process of Skolemization [Chan1987]. Such methods focus on the translation process in order to utilize ordinary propositional resolution theorem provers.

In this paper however, we show that we can directly extend propositional resolution to modal clauses. Our system is based on the sound rules of inferences for modal operators which have been shown to be correct and complete for the KD modal logic [Enjalbert and del Cerro1989]. We present an algorithm titled ZIPPER, that applies these rules in a recursive fashion to (possibly nested) modal-propositional clauses. The benefit to this solution is that it does not require any complex transformations (e.g SNF) that expand the size of the input knowledge base. Furthermore, it can take advantage of well-known heuristics developed for traditional resolution. Ultimately, we implemented ZIPPER

in a fully functioning resolution-based theorem-proving system called EpiRes that uses linear resolution to resolve modal propositional clauses.

1.4 Overview

Given the initial implementation of EpiRes, this paper specifically seeks to validate the ZIPPER algorithm and contribute to the overall quality of EpiRes via debugging and heuristic improvements. In order to do so, we begin by implementing the classic muddy children problem [Fagin *et al.*1995] as a simple benchmark for correctness. Such a problem is apt for EpiRes as it illustrates the value gained by reasoning about others' beliefs to make inferences. We consider varying scenarios of three children and provide detailed reasoning chains, generated by EpiRes, to rigorously prove queries made.

We then test EpiRes with a more complicated example dubbed "the reactor domain" [Zhang *et al.*2018]. This example was originally crafted to highlight the dynamics of privacy preserving planning algorithms, but it is also well suited for our application. Naturally, such domains are concerned with managing the knowledge/beliefs states of various agents (what is or is not allowed to be known by a particular agent) which can be written intuitively in EpiRes.

We also present a new "meta" operator called "knows whether" (KW) which encompasses the expression of knowing a fact despite its true value being true or false. The value of such an operator is that it is easy to express that an agent knows *something*, but is agnostic to what the actual value is.

This paper will begin by describing the syntax and semantics of the modal language used, structure and usage of modal clauses, and formally present the ZIPPER algorithm. Next, we present the applications to the muddy children and reactor domains including several non-trivial queries and the introduction of the KW operator. Finally, we discuss the progress made, current limitations, and plans for future work.

2. FORMALISM

In this section, we describe the syntax and semantics of the logic KD (a subset of KD45) as our propositional multi-modal logic for representing beliefs of agents.

2.1 Syntax

Our language $\mathcal{L}$ consists of a set of propositional symbols, $\mathcal{P}$, extended with a set of modal operators $\Box_i$ and $\Diamond_i$ indexed by agent $i \in \{1 \dots n\}$, with the following recursive definition:

- if $\phi \in \mathcal{P}$ then $\phi \in \mathcal{L}$;
- if $\phi \in \mathcal{L}$ then $\neg\phi, (\phi) \in \mathcal{L}$;
- if $\phi, \psi \in \mathcal{L}$ then $\phi \vee \psi \in \mathcal{L}$;
- if $\phi \in \mathcal{L}$ then $\Box_i\phi \in \mathcal{L}$ and $\Diamond_i\phi \in \mathcal{L}$.

This allows arbitrarily nested modal-propositional sentences. Additional Boolean connectives like $\wedge$ and $\rightarrow$ are easily defined using the simple recursive definition. We will use $\perp$ to denote false (or the empty clause) (so $\perp \in \mathcal{L}$). Traditionally, $\Box\phi$ is interpreted as "necessarily ϕ," but in a multi-agent context, we will interpret $\Box_i\phi$ as "agent i believes ϕ." The semantics supporting this is given below. Similarly, $\Diamond_i\phi$ will be interpreted as "agent i believes ϕ is possible." As noted below, $\Diamond_i\phi \equiv \neg\Box_i\neg\phi$.

2.2 Semantics

The semantics of our multi-modal language are defined using Kripke-style Possible Worlds semantics. We assume there is a set of possible worlds $\mathcal{W}$ each of which determines truth assignments over the set of propositions $\mathcal{P}$ (using $f : \mathcal{P} \times \mathcal{W} \rightarrow \{T, F\}$). Furthermore, let $w_0 \in \mathcal{W}$ be a designated world representing the actual current state, and let $\sim_1 \cdots \sim_i \cdots \sim_n$, with each $\sim_i \subseteq \mathcal{W} \times \mathcal{W}$, be a set of "accessibility" relations among possible worlds.

A model M is a triple $\langle \mathcal{W}, f, \{\sim_1 \cdots \sim_n\}\rangle$. Then truth conditions are defined by these rules ($\models$ is read as "satisfies"):

- $M, w_0 \models \phi$ for $\phi \in \mathcal{P}$ iff $w_0 \models \phi$ or $f(\phi, w_0) = T$;
- $M, w_0 \models \neg\phi$ iff it is not the case that $M, w_0 \models \phi$ or truth assignment of ϕ is false in M, w_0 ($f(\phi, w_0) = F$);
- $M, w_0 \models \phi \vee \psi$ iff $M, w_0 \models \phi$ or $M, w_0 \models \psi$;
- $M, w_0 \models \Box_i\phi$ iff for all $w \in \mathcal{W}$ such that $w_0 \sim_i w$, they have $M, w \models \phi$;
- $M, w_0 \models \Diamond_i\phi$ iff there exists $w \in \mathcal{W}$ such that $w_0 \sim_i w$ and $M, w \models \phi$.

The last definition can be used to prove $\Diamond_i\phi \equiv \neg\Box_i\neg\phi$ based on the accessibility relation over possible worlds.

2.3 Axioms for KD

KD logic is a subset of KD45, which is often taken as a reasonable approximation for representing agents' beliefs. KD logic is characterized by these axioms:

(NEC)	ϕ	$\rightarrow \Box\phi$	(necessitation)
(K)	$\Box(\phi \rightarrow \psi)$	$\rightarrow (\Box\phi \rightarrow \Box\psi)$	
(D)	$\Box\phi$	$\rightarrow \Diamond\phi$	(consistency)

KD logic eschews the positive and negative introspection axioms of KD45 (axiom (4): $\Box\phi \rightarrow \Box\Box\phi$, axiom (5): $\neg\Box\phi \rightarrow \Box\neg\Box\phi$), but nonetheless is still a useful system for modeling agents' beliefs, as we demonstrate with the Muddy Children problem in Chapter 4. Whereas KD45 axioms taken together imply the constraint that the accessibility relations among possible worlds are transitive, serial (also known as total), and Euclidean, axioms K and D alone only require the accessibility relations to be serial ($\forall u \in \mathcal{W}$, $\exists v$ s.t. $u \sim_i v$). The T (Truth) axiom ($\Box\phi \rightarrow \phi$) is not included here because, for multi-agent applications, we want to allow some agents to possibly have inaccurate beliefs. Nonetheless, each agent's beliefs

must be self-consistent (non-contradictory), enforced by axiom (D) (there must always exist at least one world an agent believes is possible in any given state). This contrasts with $\mathcal{VSK}$ logic, where the axiomatization based on S5 logic requires consistency with the actual world [Wooldridge and Lomuscio2000]. We will use 'believes' and 'knows' interchangeably in this paper.

3. ALGORITHM FOR RESOLUTION OF MODAL CLAUSES

In this section, we describe an algorithm that instantiates the modal resolution rules described above, called ZIPPER. The ZIPPER algorithm is incorporated in an automated deductgion system called EpiRes (for Epistemic Reasoner). Given a knowledge base with modal clauses (written in an ASCII syntax) and a query sentence, EpiRes converts the sentences to CNF, iteratively resolves pairs of sentences until it generates the empty clause (if the query is entailed), and then prints out a proof tree.

3.1 Representation of Modal Clauses

To describe the algorithm, it is helpful to choose a representation of modal clauses (mclause) that conveniently matches the hierarchical syntactic structure. A regular propositional clause can be thought of in prefix notation as an $\vee$ operator followed by a list of literals, $L_1 \ldots L_n$. Even a single literal by itself can be thought of as a unit clause ($\vee$ operator with a list of length 1, e.g. $q \equiv$`(or q)`). When $n = 0$, it represents the empty clause. Modal clauses simply extend the list representation with modal operators $\Box_i$ and $\Diamond_i$. Both $\Box_i$ and $\Diamond_i$ operators may have a list of mclauses, which are implicitly treated as a conjunction. Due to its relevance in the algorithm below, we note that $\Box_i$ operators are constrained to have only a single mclause argument, whereas, $\Diamond_i$ operators can have multiple mclauses (jointly defining a possible-world context). Hence the BNF grammar for the mclause representation is:

⟨mclause⟩ ::= `(or` ⟨oclause⟩∗`)`

⟨oclause⟩ ::= ⟨literal⟩

| ⟨koper⟩ ⟨mclause⟩

| ⟨poper⟩ ⟨mclause⟩ ⟨mclause⟩∗

⟨literal⟩ ::= ⟨prop⟩ |`(not` ⟨prop⟩`)`

⟨koper⟩ ::= `[1]` | `[2]` | `[3]` | ···

⟨poper⟩ ::= `<1>` | `<2>` | `<3>` | ···

Note that "`[i]`" and "`<i>`" are the ASCII renderings of $\Box_i$ and $\Diamond_i$, respectively, for agent index i.

The representation of an mclause maps onto an Abstract Syntax Tree (AST) in a natural way. The leaves are literals, and the internal nodes are one of the 3 types of operators, $\vee$, $\Box$-type, and $\Diamond$-type. The children of an operator node consist of the list of arguments, which are literals (leaves) or nested mclauses (sub-trees).

3.2 Conversion to Modal-CNF

Rather than having to write domain knowledge in CNF, it is more convenient to express knowledge in the form of modal-propositional sentences, which has a more flexible syntax and adds operators (connectives) such as $\rightarrow$ and $\wedge$. KB's for EpiRes can be written as general propositional sentences (similar to the ASCII prefix-notation syntax described above, augmented with operators '`->`' and '`and`'), such as `(-> (or a (not b)) (and c ([i] c)))`, meaning: $(a \vee \neg b) \rightarrow (c \wedge \Box_i(c))$. Fortunately, a KB written in modal-propositional logic can easily be converted to CNF as a pre-processing step. (The example above yields 4 modal-clauses.) The same transformations used for converting classical propositional sentences to CNF, such as implication elimination, pushing negations inward, and distribution, are applied recursively within modal sub-expressions as well, resulting in a conjunction of clauses at the top level containing literals or nested modal-clause sub-expressions. Each $\Box_i$ operator (at any level) has a (possibly modal) clause as an argument, and each $\Diamond_i$ operator can have multiple clauses (defining a possible world). While pushing negations inward over modal operators ($\Box$ and $\Diamond$), we flip the operator using $\neg\Box_i\phi \equiv \Diamond_i\neg\phi$, and its dual.

The CNF-conversion process also 'flattens' instances of like-operators, such as *and*-over-*and* and *or*-over-*or*. So `(and p (and q r) (or s t (or u w) (or)) (or x))` is converted to `(and p q r (or s t u w) x)`. Note that one of the disjuncts was an empty clause and thus was dropped (application of simplification rule S_2).

The process of converting a modal-propositional KB into modal-CNF ensures that the input to EpiRes consists of modal clauses adhering to the syntax and hierarchical repre-

sentation described above, on which ZIPPER relies.

3.3 Modal Contexts and Branching Sequences

Next, we introduce two useful concepts to describe syntactic context. For any node in the AST of a modal clause, the *Modal Context* is the list of operators on the path from the root down to the node. This will be an alternating list of $\vee$ operators and modal operators, $\Box_i$ or $\Diamond_i$.

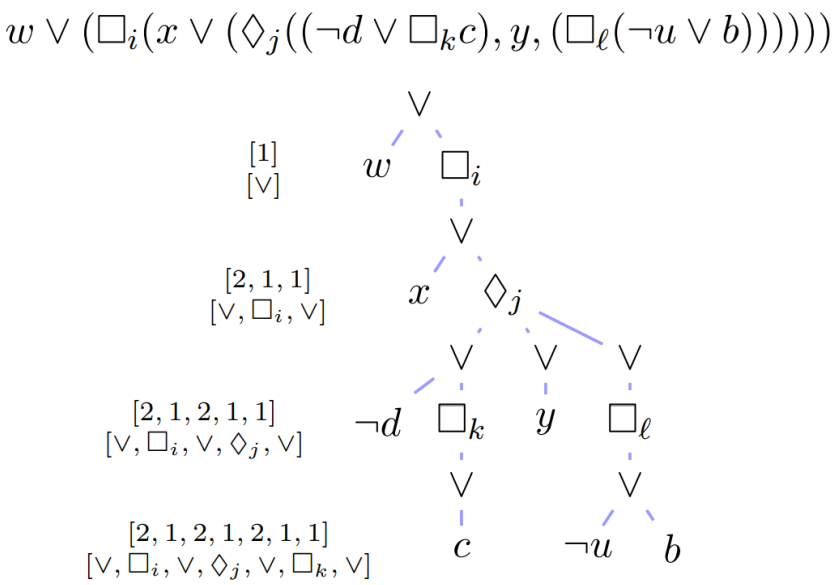

Figure 1: Parse tree of a modal clause, showing the Branch Sequences and Modal Contexts for literals: w, x, $\neg d$, and c.

In $w \vee (\Box_i(x \vee (\Diamond_j((\neg d \vee \Box_k c), y, (\Box_\ell(\neg u \vee b))))))$, for example, literal c has the modal context of $[\vee, \Box_i, \vee, \Diamond_j, \vee, \Box_k, \vee]$ (the c is implicitly in a clause of length 1), whereas the context of $\neg d$ is $[\vee, \Box_i, \vee]$ and w is $[\vee]$ (top-level disjunct). This is illustrated in Figure 1.

The *Branching Sequence* is a list of integers describing the path to a node by specifying which child in the list of arguments for each nested operator represents a branch on the path (integers, starting with 1). For example, in the modal clauses above, c has the branching sequence of $[2, 1, 2, 1, 2, 1, 1]$, whereas $\neg d$ is $[2, 1, 1]$ and w is $[1]$ (see Figure 1).

Definition 1. *Two modal contexts $U_1 \ldots U_n$ and $V_1 \ldots V_n$ are defined to* match *iff: 1) they have the same length, 2) corresponding operators (U_i and V_i for each i) are either both $\vee$, both $\Box$, or one $\Box$ and one $\Diamond$ (but not both $\Diamond$), and 3) the agent indexes of the correponding modal operators match.*

For example, if $U_3 = \Box_b$ (3 levels deep), then V_3 must be $\Box_b$ or $\Diamond_b$ (both for agent b).

The algorithms for extracting the Modal Context and Branching Sequence down to each literal, as well as determining whether the Modal Contexts for literals in two expressions match, are straightforward (e.g. using recursion) and run in linear-time in the size of the expression.

3.4 The ZIPPER algorithm for Modal Resolution

The algorithm for resolving two modal clauses has 3 steps:

1. Extract the Modal Contexts of each literal in both expressions.
2. Find a pair of literals (one in each parse tree) that are resolvable (complementary and in matching modal contexts).
3. Combine the two clauses using a recursive procedure called 'zippering'.

The third step is called the ZIPPER algorithm because of how it merges sub-clauses at each level of the two expressions bottom-up. For two resolvable literals, at the bottom of each path are complementary literals that can be cancelled out (forming the empty clauses). At the level above in the operator paths involving $\vee$ operators), the children are concatenated (excluding the complementary literals), analogous to combining literals in propositional resolution. At levels further up in the operator paths, nodes with matching $\Box$ operators, or one $\Box$ and one $\Diamond$, may be combined according to the rules defined by [Enjalbert and del Cerro1989]. By the time the recursive process reaches back up to the top level, a single combined clause is produced as the resolvent.

Critically, the hierarchical structure of mclauses is preserved by the ZIPPER algorithm. Resolvents generated by ZIPPER also have an AST with flattened clauses (disjuncts of literals or modal sub-expressions) at each level (except $\Diamond$ operators, which can have multiple clauses). Each path from a root node down to a literal (leaf node) will have sequence of operators alternating between $\vee$ (including first and last) and modal operators (type $\Box$ or $\Diamond$).

An important part of the algorithm is that empty clauses nested in $\Box$ modal contexts are propagated upward, according to Simplification Rules S_2 and S_Q. Hence if a contradiction is discovered in the context of the beliefs of another agent (or beliefs of one agent about another), then this constitutes a contradiction, showing the query was indeed entailed. (Note that the parent of a $\Box$ operator will be an $\vee$ operator, and an empty clause as a disjunct will get dropped if there are other disjuncts remaining (simplification rule S_2), or will propagate further upward as an empty clause (rule S_Q).)

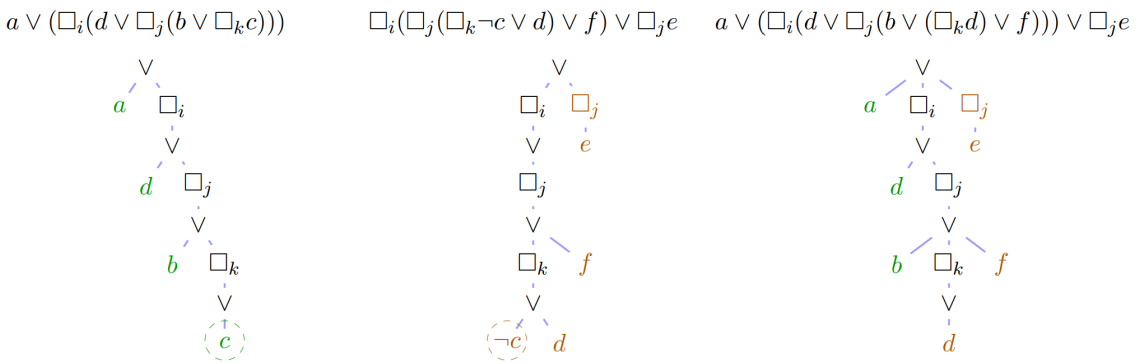

Figure 2: Parse tree of a modal clause, showing the Branch Sequences and Modal Contexts for literals: w, x, $\neg d$, and c.

Figure 2 illustrates the zippering process to resolve two modal clauses. Note: the modal contexts of the selected complementary literals *match* (paths to leaf nodes c and $\neg c$): $[\vee, \Box_i, \vee, \Box_j, \vee, \Box_k, \vee]$.

3.5 Implementation of Resolution in EpiRes

EpiRes uses *Linear Resolution* as a search strategy, meaning that two clauses are resolved only if one is an input clause, or one is an ancestor of the other. Linear Resolution is complete for refutation proofs [Loveland1970, Luckham1970]. Pairs of resolvable clauses satisfying this criterion are placed in a priority queue to be resolved using the ZIPPER algorithm.

EpiRes provides several different heuristics for prioritizing clause-pairs to select from the queue. One is an extension of the classical Unit-Clause preference, where clause-pairs are prioritized by the minimum of the lengths of the 2 clauses (which aides in generating shorter resolvents, in search of the empty clause). However, the classic notion of clause length, which is evaluated only on the number of disjuncts at the top-level, can be extended and customized for modal clauses. Heuristics adapted for modal clauses are to prioritize clause-pairs by the minimum of the tree (AST) sizes or total number of literals at any level (leaves), which are easily calculated by depth-first traversal. For example, $r \vee \Box_a(\Box_b(s \vee t) \vee \neg w)$ has 2 top-level disjuncts, but 4 literals overall (as leaves). In practice, the heuristic that has worked the best is to score clause-pairs by the minimum number of literals in the two clauses plus 0.1 times the max number of literals, but EpiRes enables other heuristics to be developed and explored for making the search for the empty clause as efficient as possible.

Importantly, EpiRes checks whether new resolvents match previously-generated clauses, and if so, discards them. This greatly reduces the queue size of resolvable pairs. To do this efficiently, clauses must be *canonicalized* by putting literals in alphabetical order, and recursively doing this within modal sub-expressions as well, so two clauses can easily be compared syntactically without worrying that they look different but are really the same. EpiRes also performs *factoring* to eliminate repeated literals in each clause (simplification rule S_4, which is applied to all resolvents generated by ZIPPER).

Algorithm 1 ZIPPER(mclauses C_1, C_2, BranchSeqns I, J) combine two clauses to produce a resolvent, based on BranchSequences representing paths to selected complementary literals in matching modal contexts.

▶ base case: leaf nodes (should be complementary literals)
if C_1 and C_2 are literals **then** return $\varnothing$ **end if**
▶ view clauses as lists with operator ρ_x first, then args $A_x \ldots$
$\rho_1 \leftarrow C_1$.head(); $A_1 \leftarrow C_1$.tail()
$\rho_2 \leftarrow C_2$.head(); $A_2 \leftarrow C_2$.tail()
$i \leftarrow I$.head(); $I' \leftarrow I$.tail()
$j \leftarrow J$.head(); $J' \leftarrow J$.tail()
if ρ_1 is a $\Diamond$ operator and ρ_2 is a $\Box$ operator **then**
  return ZIPPER(C_2, C_1, J, I) ▶ swap order
end if
▶ Z is resolvent of sub-expressions on path (could be $\varnothing$)
$Z \leftarrow$ ZIPPER$(A_1[i], A_2[j], I', J')$
switch (ρ_1, ρ_2) **do**
  ▶ $\vee$-rule
  case $(\vee, \vee)$
    $D \leftarrow A_1[-i] \cup A_2[-j] \cup \{Z\}$ ▶ drop compl. literals
    if $D = \varnothing$ **then** return $\varnothing$ **end if** ▶ empty clause
    return Expr$(\vee, D) = D_1 \vee \cdots \vee D_n$
  ▶ since $\Box$ operators have 1 arg each, Z replaces A_1 and A_2
  case $(\Box, \Box)$
    if $Z = \varnothing$ **then** return $\varnothing$ **end if** ▶ S_Q rule: $\Box\bot \rightarrow \bot$
    return Expr(ρ_1, Z) $= \Box_i(Z)$ ▶ $\Box\Box$-rule
  ▶ simplification rules: (S_1) $\Diamond\bot \rightarrow \bot$; (S_3) $C, \bot \rightarrow \bot$
  case $(\Box, \Diamond)$
    $\alpha = A_2 \cup \{Z\}$ ▶ $\Box\Diamond$-rule adds new resolvent Z to previous clauses of $\Diamond$ context
    $\beta \leftarrow \mathit{ResolutionClosure}(\alpha)$ ▶ Γ rule $\Diamond$-rule1: expand clauses in the new $\Diamond$ context
    if $\bot \in \beta$ **then** return $\varnothing$ **end if** ▶ if clauses are inconsistent, collapse to empty clause and propagate up
    return Expr$(\rho_2, \beta) = \Diamond_i(\beta) = \Diamond_i(RC(A_2 \cup \{Z\}))$
raise error ▶ should never be reached if clauses are resolvable

4. APPLICATION: MUDDY CHILDREN

The first example we consider in demonstrating the utility and expressiveness of EpiRes is the widely-used Muddy Children problem [Fagin *et al.*1995]. In this scenario, we consider a world in which several children may or may not have mud on their forehead – a location visible to others, however not to themselves. The teacher begins by announcing to all of the children that at least one of them has a muddy forehead and proceeds to ask all children who know they have muddy foreheads to raise their hand. In the case where two or more children have mud, nobody responds upon the first asking of the question. The teacher will continue asking the exact question for multiple rounds until all children are able to correctly raise their hand. Given $M \geq 2$ muddy children, this final round occurs on the Mth time (assuming of course that all children are properly perceptive and make perfectly rational decisions).

We will instantiate this example in multiple scenarios involving three children in which the number of muddy children vary. We begin with children a, b, and c who are all gathered together in the presence of the teacher. We omit the example of zero muddy children as this is trivial; rather we start by examining scenario 1 where only child a is muddy – written as the proposition M_a. Subsequently in scenario 2, both child a and child b will be muddy.

4.1 Running EpiRes

4.1.1 Constructing the Knowledge Base

For this example (and subsequent examples) there will be several lines of modal-propositional sentences as well as ASCII representations of the sentences in CNF form denoted by the monospace font. Conjunctive Normal Form can accurately represent any modal-propositional sentence by converting it into another logically equivalent conjunction of clauses in which 'and's are distributed over 'or's (i.e $(A \vee B) \wedge (C \vee D)$ for arbitrary propositions A,B,C,D).

Although EpiRes requires the input knowledge base to be in CNF form, it is unnatural for human to think in such a way. For example, is it more intuitive to consider $A \rightarrow B$ rather than its equivalent in CNF form $\neg A \vee B$. For this reason, we developed an additional python script to take such KBs written in the higher level modal-propositional logic – including symbols such as $\wedge$, $\rightarrow$, $\leftrightarrow$, etc. into the CNF format.

These modal-propositional sentences are written with propositions (such as `Ma` in this example) as well as logical and modal operators in prefix notation. The modal operators $\Box_i$ will be written as `[i]` and $\Diamond_i$ as `<i>` for an agent i.

4.1.2 Testing a Query

After the knowledge base has been constructed and translated into the proper CNF form, EpiRes is ready to test queries against it to determine if a modal-propositional can be entailed.

EpiRes mimics the structure of a typical resolution theorem prover such that to show a query is entailed, the query is negated, added to the KB, and the prover is ran until the empty clause is derived thus signaling it is inconsistent to believe the input KB and the negated query; thus, it must be the case that original query is true. At this point EpiRes generates and prints the resulting proof tree with the empty clause at the highest level.

EpiRes will continue to run until either the empty clause is found or there are no longer any remaining pairs that can be resolved together. For the latter case, note that this does *not* mean that the query is not entailed (more on this in Section 6.2). For the sake of this paper, EpiRes was often terminated manually after 100,000 iterations.

4.2 Scenario 1:

For the first scenario, we consider the case where only one of three children are muddy, namely M_a,$\neg M_b$,$\neg M_c$.

The teacher's announcement that at least one of them is muddy can be represented by

$$\Box_i(M_a \vee M_b \vee M_c) \text{ for } i \in \{a, b, c\}$$

In EpiRes, these sentences are encoded as:

```
([a] (or Ma Mb Mc))
([b] (or Ma Mb Mc))
([c] (or Ma Mb Mc))
```

Following this problem through the perspective of child b, we can represent the observations that he sees:

$$M_a \rightarrow \Box_b M_a,\ \neg M_c \rightarrow \Box_b \neg M_c$$

```
([b] Ma)
([b] (not Mc))
```

Another group of important rules to consider is the notion of visibility. Here we venture one modal level deeper as we consider the case where one entity reasons about the beliefs of another. Crucial to solving this problem, we capture what child b can reason about what the other two children are able to perceive. That is, if child b was muddy (or not), both children a and c would be able to see it (or not). In other words, these sentences capture the notion that each child knows that every other child is aware of the previous rules. That is,

$$\Box_b(M_a \rightarrow \Box_b M_a),\ \Box_b(\neg M_c \rightarrow \Box_b \neg M_c), \text{ etc.}$$

```
([b] (or (not Mb) ([a] Mb)))
([b] (or Mb ([a] (not Mb))))
([b] (or Mc ([a] (not Mc))))
([b] (or (not Mc) ([a] Mc)))
```

In this particular example, both children a and c are able to perceive that child b does not have a muddy forehead, but it is important to consider both cases since this is unknown to b a priori.

In this scenario, a will be able to infer he is muddy in the initial round (because he sees that none of the other children have mud), however b is unsure about himself because he sees mud on a's forehead, and this could be sufficient to satisfy the teacher's announcement. (Child c follows the same reasoning of child b independently).

4.2.1 Iteration 0

Child a raises his hand, however the other children are unsure whether their foreheads are muddy or not. That is, $\Box_a M_a$ is entailed, however $\Box_b \neg M_b$ and $\Box_c \neg M_c$ are not entailed.

The reasoning from the perspective of child a is simple. He knows that at least one child has a muddy forehead, yet he perceives that children b and c do not have muddy foreheads thus he must have a muddy forehead. This can be proved by EpiRes.

Recall that since EpiRes is a resolution based theorem prover, we show that a query is entailed by asserting the negated query and subsequently reaching the empty clause. Thus in order to prove $\Box_a M_a$, we assert $\neg(\Box_a M_a)$ which translates to $\Diamond_a \neg M_a$ (by pushing the negation inward and flipping the operator).

Table 1: Scenario 1 initial knowledge base

Sentence	**Annotation**
`0. (or (a (or Ma Mb Mc)))`	Announcement
`1. (or (a (or (not Mb))))`	Visible
`2. (or (a (or (not Mc))))`	Visible
`3. (or (a (or (not Ma))))`	Negated query

The following is the resulting proof tree generated after EpiRes is executed with the initial knowledge based described in Table 1

```
...success, found the empty clause
6. (or)
  5. (or ([a] (or Ma)))
    4. (or ([a] (or Ma Mc)))
      0. (or ([a] (or Ma Mb Mc)))
      1. (or ([a] (or (not Mb))))
    2. (or ([a] (or (not Mc))))
  3. (or (<a> (or (not Ma))))
```

When EpiRes succeeds in generating the empty clause, it prints out a hierarchical representation of the proof tree, showing the previous clauses from which each clause was derived (with the empty clauses at the root). Since the initial knowledge base contained four clauses, note that clauses 0-3 are from the input knowledge base while clauses 4-6 are derived by EpiRes during runtime.

The empty clause (6) is generated by sentences 5 and 3 which were combined using Enjalbert's $\Box\Diamond$-rule to resolve opposing literals within the $\Box$ and $\Diamond$ operators respectively. Similarly, sentences 4 and 2 were combined to produce sentence 5 and sentences 0 and 1 were combined to produce sentence 4 both using Enjalbert's $\Box\Box$-rule which defines resolution within the modal context of the $\Box$ operator.

Additionally, you can consider the proof tree semantically:

```
6. contradiction
  5. a knows that he must be muddy
    4. a knows that himself or c is muddy
      0. a knows that at least someone is muddy
      1. a knows b is not muddy
    2. a knows c is not muddy
  3. a believes it is possible for himself to not be muddy
```

Finally, consider the following diagram which visually depicts the proof of $\Box_a M_a$. Literals to be resolved are bolded and the clause with the positive literal is shaded blue while the clause with the negative literal is shaded red.

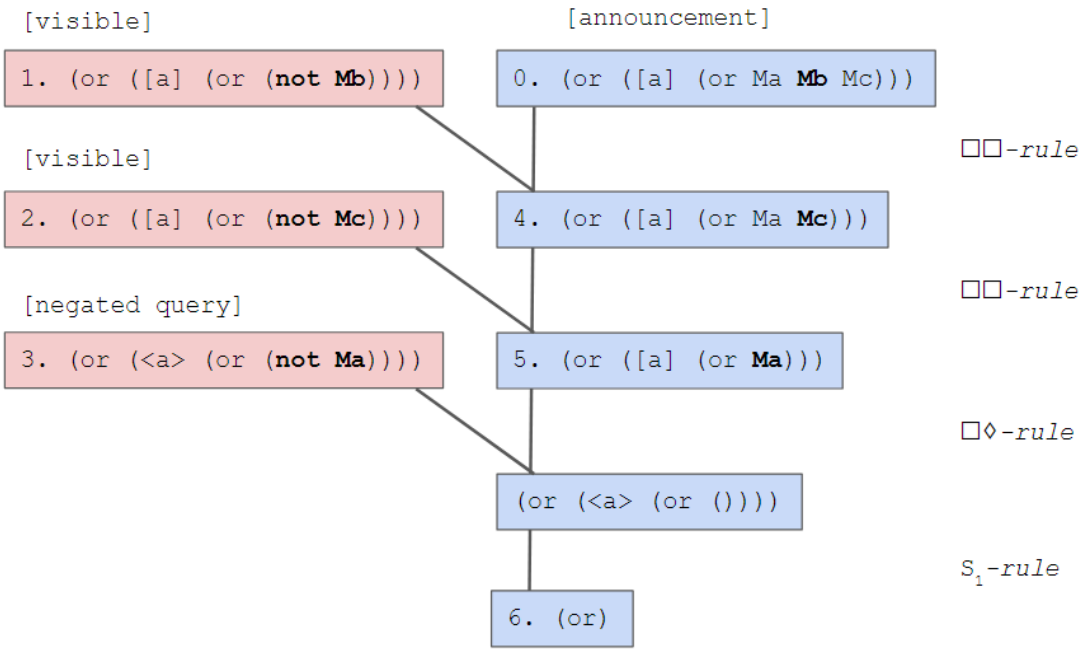

Figure 3: Scenario 1 proof tree showing that, a knows a is muddy

Note the linear structure of the proof which EpiRes discovered to be a pattern in several subsequent queries that were tested. This led to an optimization in which we prioritized resolving clauses that were either in the original knowledge base input or were ancestors of one another.

Now consider the perspective of child b. He perceives that child a has a muddy forehead and that child c does not have a muddy forehead. Since the teacher's announcement explicitly allows for one or more children to have muddy foreheads, child b cannot confidently assert that he is muddy or not. This is represented as $\neg(\Box_b M_b)$.

Note that this cannot be proved using EpiRes. This is not a limitation of EpiRes

in particular, but rather a limitation of resolution refutation theorem provers in general; such systems are unable to prove queries that are not entailed from the knowledge base. If this is negated and added to the knowledge base, EpiRes "fails" in that it will not produce the empty clause. In this simple domain, EpiRes actually halts because it generates all the resolvents that are possible from the input clauses (exhausts the queue of resolvable pairs; this requires factoring and canonicalization to detect when a newly-derived clause matches a previous clause, and can thus be ignored); in more complex problems, EpirRes keeps running without generating the empty clause, and it has to be terminated prematurely, which makes the answer ambiguous (as to whether the query is entailed). Yet, there is no direct way to show that something is not entailed using resolution refutation.

Luckily in our case, this example is simple enough to reason logically why such a statement is true. Since child b sees mud on a's forehead, it would be consistent with the teachers assertion either way, whether he had mud or not; neither state is entailed, from b's perspective. We can also ask whether b knows that he is not muddy in iteration 1 after he sees a say that he knows a is muddy. It seems deceptively obvious: b should be able to infer that b is not muddy because, if they were, then a would not have been able to determine a was muddy. However, this is more difficult to prove because it requires abductive reasoning (i.e. what would have to be true to make the query entailed?).

Moving on to the next scenario, we now consider children a, b, and c again, however this time child a and child b are both muddy (i.e M_a and M_b).

4.3 Scenario 2

For this scenario (M_a,M_b,$\neg M_c$), much of the knowledge base remains consistent. The teacher's announcement is encoded the same as:

$$\Box_i(M_a \vee M_b \vee M_c) \text{ for } i \in \{a, b, c\}$$

```
([a] (or Ma Mb Mc))
([b] (or Ma Mb Mc))
([c] (or Ma Mb Mc))
```

It will later become apparent that in order to prove $\Box_b M_b$, it is necessary to encode that child b knows that the other children also heard the teacher's announcement. In this proof, we only need child b's reasoning about child a:

$$\Box_b \Box_a (M_a \vee M_b \vee M_c)$$

```
([b] (or ([a] (or Ma Mb Mc))))
```

Again we follow the perspective of child b in which he sees:

$$M_a \rightarrow \Box_b M_a,\ \neg M_c \rightarrow \Box_b \neg M_c$$

```
([b] Ma)
```

```
([b] (not Mc))
```

Again, we also follow the same visibility rules,

$$\Box_b (M_a \rightarrow \Box_b M_a),\ \Box_b (\neg M_c \rightarrow \Box_b \neg M_c),\ \text{etc.}$$

```
([b] (or (not Mb) ([a] Mb)))
```

```
([b] (or Mb ([a] (not Mb))))
```

```
([b] (or Mc ([a] (not Mc))))
```

```
([b] (or (not Mc) ([a] Mc)))
```

4.3.1 Iteration 0

All children keep their hands lowered since they are unsure if they are muddy or not (i.e both $\Box_a M_a$ and $\Box_b M_b$ fail as queries), since each child sees mud on at least one other child's forehead.

This follows from the same reasoning as the previous example; children a and b both see one child with a muddy forehead and thus are unsure about their own forehead since that would be sufficient to satisfy the teacher's announcement. Similarly, child c sees the other two children with muddy foreheads and for the same reason he does not know for certain the status of his own forehead.

In the next iteration, child b now knows that child a does not know that he is muddy since child b observed that child a did not raise his hand. Note that the same could be said about child b's reasoning about the belief state of child c, however this is not needed in this particular example. The knowledge base now grows with this added information; we add the assertion that child b knows child a knows a is not muddy:

$$\Box_b(\neg(\Box_a M_a))$$

```
([b] (not ([a] Ma)))
```

4.3.2 Iteration 1

Child b raises his hand signaling that he knows that he has a muddy forehead (i.e $\Box_b M_b$ succeeds). A similar line of reasoning could be shown for child a, however this concludes the example as it pertains to following child b.

The following statements summarize the above observations, visibility rules, and negated query followed by a proof tree detailing the steps to reach the empty clause.

Table 2: Scenario 2 Initial knowledge base

Sentence	**Annotation**
`0. (or ([a] (or Ma Mb Mc)))`	Announcement
`1. (or ([b] (or Ma Mb Mc)))`	Announcement
`2. (or ([c] (or Ma Mb Mc)))`	Announcement
`3. (or ([b] (or ([a] (or Ma Mb Mc)))))`	b knows a heard the announcement
`4. (or ([b] (or Ma)))`	Visible
`5. (or ([b] (or (not Mc))))`	Visible
`6. (or ([b] (or ([a] (or Mb)) (not Mb))))`	Visibility rules
`7. (or ([b] (or ([a] (or (not Mb))) Mb)))`	Visibility rules
`8. (or ([b] (or ([a] (or (not Mc))) Mc)))`	Visibility rules
`9. (or ([b] (or ([a] (or Mc)) (not Mc))))`	Visibility rules
`10. (or ([b] (or (<a> (or (not Ma))))))`	Result of iteration 0
`11. (or (<b> (or (not Mb))))`	Negated query

The following is the resulting proof tree generated after EpiRes is executed with the initial knowledge based described in Table 2

```
 ...success, found the empty clause
23. (or)
  22. (or ([b] (or Mb)))
    21. (or ([b] (or ([a] (or Ma)) Mb)))
      18. (or ([b] (or ([a] (or Ma Mb)))))
        12. (or ([b] (or ([a] (or (not Mc))))))
          5. (or ([b] (or (not Mc))))
          8. (or ([b] (or ([a] (or (not Mc))) Mc)))
        3. (or ([b] (or ([a] (or Ma Mb Mc)))))
      7. (or ([b] (or ([a] (or (not Mb))) Mb)))
    10. (or ([b] (or (<a> (or (not Ma))))))
  11. (or (<b> (or (not Mb))))
```

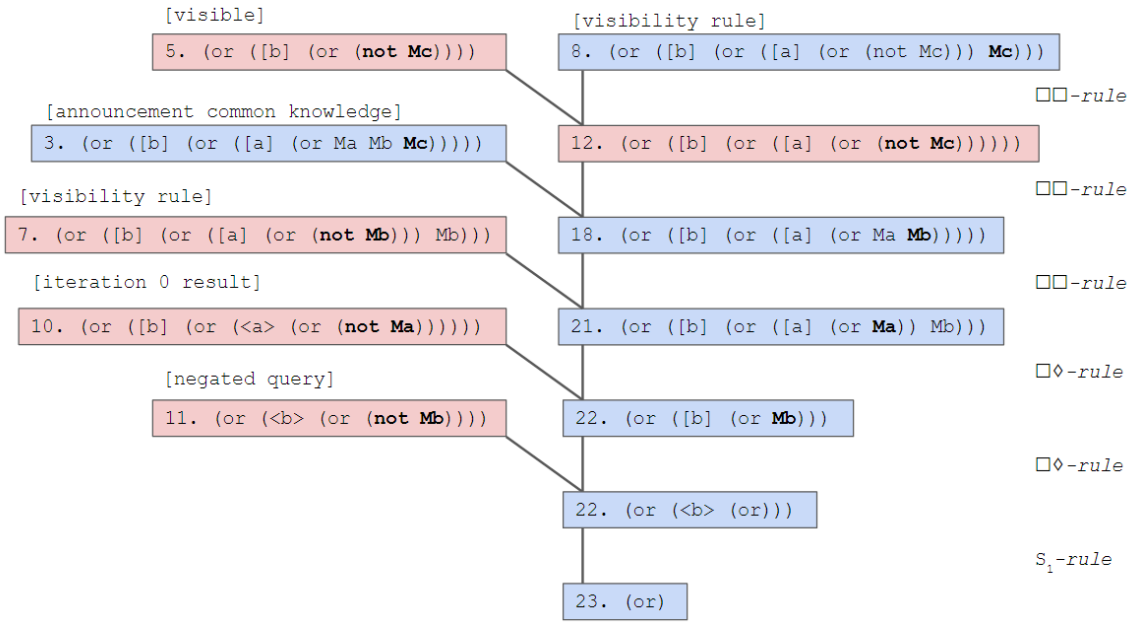

Figure 4: Scenario 2 proof tree showing that, after b hears a does not know whether a is muddy, b now knows b is muddy.

For the above proof tree, there is an additional step (not shown). Clauses 22 and 11 initially resolve to give $\Diamond_b \bot$ which has an empty clause embedded in the modal context $\Diamond_b$. But applying a simplification rule in [Enjalbert and del Cerro1989] ($\Diamond \bot \rightarrow \bot$), this then propagates upward to make the whole clause empty at the top level.

This is an important example to explain the chain of epistemic reasoning, as this is a non-trivial conclusion that highlights the power of EpiRes. One way to analyze this proof is to consider the various possible worlds child b considers. Let's begin by viewing all possible cases:

$$\neg a, \neg b, \neg c$$

$$\neg a, \neg b, c$$

$$\neg a, b, \neg c$$

$$\neg a, b, c$$

$$a, \neg b, \neg c$$

$$a, \neg b, c$$

$$a, b, \neg c$$

$$a, b, c$$

Immediately, we can eliminate the world in which all children are clean since this contradicts the teacher's announcement. Also, we reduce the possible worlds to only those which are consistent with what child b perceives (that is a and $\neg c$). Note that the only difference between the two remaining possible worlds is the status of b which is currently unknown to him.

$$a, \neg b, \neg c$$

$$a, b, \neg c$$

As previously mentioned, in iteration 0, child b is unsure if he has a muddy forehead since he sees one clean and one muddy forehead. Hence, the key to child b reaching his

conclusion is contingent on the results of iteration 0: nobody is sure if they have mud on their own forehead since each child sees mud on another.

Moreover, let us consider what child b is able to reason about child a's belief state. Child b understands that both he and child a perceive child c to not have a muddy forehead. Child b cleverly knows that in the case where he did not have a muddy forehead, child a would have been able to confidently raise his hand in iteration 0 (for the same reasoning explained in the previous example). This eliminates the other possible world leaving the true state of the world as:

$$a, b, \neg c$$

In the resolution proof, the critical steps are combining clause 18 (b knows a knows M_a or M_b) with the input with the visibility rule 7 (b knows that a would know M_b if b had mud) to give 21 (b knows that either a knows a is muddy, or b has mud). 21 is then combined with the input clause representing that b knows a knows a is not muddy, because a was silent in the first round), to allow b to conclude b is muddy (22). Now during iteration 1, child b can confidently claim that he must have a muddy forehead thus ending the puzzle.

4.4 Extension to more children

This seemingly simple example articulates the unapparent intricacies in properly modeling a situation in epistemic logic for EpiRes to intelligently solve, yet the MC problem does not end here; It has been proved that given n muddy children, all children will be able to answer the teacher properly after exactly n iterations.

Although true in theory, the KB would grow exponentially has we would need to continue deeper nesting of the teacher's announcement to all children $\Box_a\Box_b\Box_c(M_a \vee M_b \vee M_c)$ for all combinations of children. This is where it becomes necessary to transition to a "common knowledge" operator such as C_G defined in RAK [Fagin *et al.*1995]. With additional syntax for such situations, EpiRes would be able to determine which children become aware they are muddy in which round (though it might take asserting several intermediate

resolution steps!). *Fortunately this is not the goal of EpiRes. The application of the MC problem to EpiRes demonstrates the power and utility of epistemic reasoning about others' knowledge/belief states under the KD modal logic.* To better handle such situations, there does exist applications in which agents can benefit by reasoning about what each other beliefs, in result of what was observed, without explicitly involving a public announcement [Baltag *et al.*1998, van Benthem *et al.*2018].

These previous examples, however, are sufficient to illustrate the ability of EpiRes to perform epistemic reasoning in multiagent situations. Furthermore, the muddy children problem shows how not only must a child reason about what he perceives, but also what he can perceive about the knowledge and beliefs of the other children; without doing so, he would not be able to come to the conclusions about himself in which he cannot observe. Additionally, the muddy children problem serves as a brief introduction to concepts such as linear resolution and Kripke world semantics. Using these important concepts as a basis, we will continue our results by examining an even more interesting application of EpiRes.

5. APPLICATION: REACTOR PROBLEM

As a motivating application, EpiRes was applied to a scenario adapted from the work of Yulin Zhang who was interested in crafting planning algorithms under the additional constraints of various information disclosure policies [Zhang *et al.*2018].

With information privacy in mind, consider a world in which a robot is tasked with inspecting various types of nuclear reactors. These facilities consist of two major reactor types: pebble or breeder. While both facilities are equipped with an entrance, exit, and a control room with a reactor type indication light, the floor plans of the two facilities differ. In breeder reactors, radiation levels can be checked at location A, and pebble reactors require radiation checks at location B – thus it is important for the robot to gather information from the indicator light in order to visit the proper location to measure radiation.

A blue light inside of the control room indicates that the reactor is of type pebble, thus the robot should move towards the bottom of the floor plan to check radiation levels at location B. On the other hand, the breeder reactor is indicated by the absence of the blue light in the control room and requires a radiation check at location A before exiting.

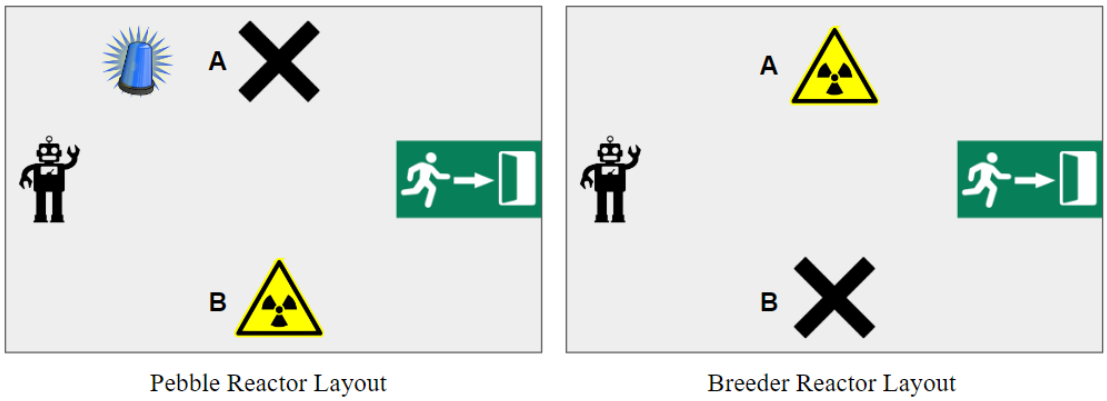

Figure 5: Reactor designs (note that the radiation symbol is the location where the robot must check, while the 'X' represents that there is nothing of interest.

As the robot is carrying out these tasks, imagine there is an observer (e.g. spy) that is monitoring the behavior of the robot. The observer seeks to determine the reactor type, which is deemed sensitive information, from the robot's actions. The observer is able to monitor the actions of the robot as it moves around the facility, however the important stipulation is that the observer must never know nor have the ability to infer the type of reactor from the information available to it. Reasoning about this problem as humans is not difficult. For example, consider a situation in which the robot enters a facility, checks the indication light, moves to location A, and finally exits. It is immediately clear that since the robot visited location A and not location B, the reactor must be of type breeder. Similarly, this reasoning can be applied to visiting location B and not A to conclude reactor type pebble. The solution to this problem (a privacy-preserving plan) is for the robot to visit both A and B; while, slightly wasteful, it masks the reactor type to an observer.

After some introspection, one can reason further to identify the inherent assumptions made to come to such conclusions. The first point to acknowledge is that we assume by the end of the robot's movements, its goals are achieved. Although this is not a valid assumption for all situations (i.e a battery dies, signals are lost, etc.) we will accept this assumption.

It is clear that the goal of the robot is to properly detect radiation (or the lack thereof) and exit the facility. The robot visiting location A and not location B does not guarantee the reactor to be of type breeder, but we assume this is true since we expect the robot to complete its goal. The reactor could have been of type pebble and thus by visiting location A, no information would be obtained and the goal would remain unachieved. Instead, we assume the goal was achieved and thus the reactor type must be breeder.

The second observation to make while introspecting is to recognize that we are able to reach a conclusion of breeder or pebble type without ever knowing if the result of the radiation test was "high" or "low." Of course, this is the information that the robot will report to the observer at the end of the testing protocol, but to us, we do not care about nor need the test results to infer the reactor type.

Since the scope of this example seeks to protect the reactor type from being disclosed, we will not need to model the state of radiation being "high" or "low." Rather we will opt to model if the robot "knows whether" radiation is "high" or "low" at a given location, which is more of an 'ambivalent' epistemic state.

The proposed planning algorithm used sets of Possible Worlds to represent the beliefs (and uncertainty) of the observer, but this approach is not scalable. Instead, we show how the beliefs of the observer and robot can be emulated using modal resolution, focusing on cases where the observer can infer (via resolution-refutation proof) the truth-value of the private facts based on observed actions of the robot, along with a knowledge base (KB) about goals and the effects of actions. After describing further details of the reactor domain, we will present the KB expressed as rules in the ASCII version of the propositional modal language.

5.1 Modeling Knows Whether

In the domain of the reactor problem, it will be useful to define a new "meta" operator of the working modal logic that we call the "knows whether" or "KW" operator for short. Recall that the observer (spy) watching the robot never obtains the radiation measurements the robot collects, but rather only views the actions that the robot took. Put another way, the spy knows that the robot knows *something*, but is agnostic to what the actual value is. Contrast this with the following statement:

$$\Box_{obs}\Box_{robot}pebble$$

This statement in words represents that the observer knows that the robot knows pebble (i.e pebble is true). Clearly this is useless as this assertion is too strong. Likewise, it would be of no meaning to consider $pebble \vee \neg pebble$ at the same nested modal level as this is always true. Ultimately, the observer is not interested in the value of $pebble$, but rather if the robot "knows whether" $pebble$.

The KW operator is a modal operator that we developed to better express the chain of reasoning for the reactor problem. For an agent to "know whether" encompasses the expression of knowing a fact despite its true value being true or false. This allows us to conveniently express the various modal states the robot will transition through as well as the robot's goal.

"Knowing whether" is not a new operator added to our formal modal logic, but rather a "meta" operator that is defined using the existing KD logic described in Chapter 2.3. It is defined under the following equivalence relationship:

$$(KWp) \equiv (p \wedge (\Box p)) \vee (\neg p \wedge (\Box \neg p))$$

For an arbitrary proposition or sub sentence p. This translation property bridges the gap between the true value of p with the beliefs state of an agent while remaining ambivalent to the actual value of p since it covers both the positive and negative case of p.

In its application to the reactor problem, a knowledge base was first written in a version of the language that augments the modal-propositional logic with the KW operator. From here, sentences were translated into the corresponding KD sentences (with only $\Box$ and $\Diamond$ modal operators) using the above equivalence and subsequently into the final conjunctive normal form (CNF) required by EpiRes.

One important note is that this translation of the KW operator strongly commits us to the notion of knowledge as "correct belief" – this corresponds to the T axiom in Kripke semantics:

$$\Box \phi \rightarrow \phi$$

This means that within the KW operator, an agent cannot "know" something that is inconsistent with the real world, however this is not true more generally in our construct since KD does not include the T axiom. Thus, it is acceptable for the robot to incorrectly believe an arbitrary p such as $\Box p$ as long as *this belief* is internally consistent.

5.2 Knowledge Base for the Reactor Inspection Domain

The following sentences are used by the observer to reason the beliefs of the robot, given its actions. They are expressed in in the ASCII representation of the modal propositional language. For simplicity, we drop the agent indexes on the modal operators, since they all refer to the epistemic state of the robot. However, since this domain involves a sequence of actions and change-of-state, we will include temporal indexes on the modal operators. We assume there are a discrete set of (totally-ordered) time-points from 0 to T, representing situations, transitioned by events or actions. Time 0 is the initial state, and time T (the finite, upper bound) is the end of an execution run (e.g. where the agent achieves its goal). We append time indexes to propositions that are fluents (whose truth-value is situation-dependent), and we append time indexes to action symbols to indicate the time at which they occur. Furthermore, we append time-indexes to modal operators to indicate what the robot knows in a given situation, (e.g. `K:t` ϕ, meaning the robot knows ϕ at time-point t). The modal operators $K_1 \ldots K_T$ are assumed to be completely independent, semantically; they have independent accessibility relations. Any connection between K operators at different time points must be constrained through rules (axioms) in the KB.

Propositions $radA$ and $radB$ will be used to represent the presence/absence of radiation at location A and B respectively. The proposition *pebble* set to True will indicate a reactor type of pebble, while False will correspond to type breeder. Additionally, the robot's actions will be encoded as propositions including: $gotoCtrlRoom$, $gotoA$, $gotoB$, $gotoExit$, (extended with time indices). Sentences in the knowledge base will be composed of modal sub sentences that leverage the $\Box$, $\Diamond$, and KW operators with various combinations of the aforementioned propositions as arguments.

Recall that in the most simple cases where the robot visits the control room, site A, and then the exit, the observer easily is able to confirm *pebble* is false, given he knows what the robot is trying to achieve; likewise for site B and *pebble* is true. Notice that in both cases, the actual value of the radiation is abstracted and not used – only the actions of the

robot were used. This is a prime candidate to implement our new KW operator.

5.2.1 Initial State

In the initial state (time-point 0), we assume the robot is at the start location, and does not know whether there is a radiation leak (at sites A or B):

```
(at_start_t0)
(not (KW:t0 radA))
(not (KW:t0 radB))
```

5.2.2 Actions

The actions of the robot depend on the plan (see Scenarios below). For example, in scenario A, the agent visits the control room, then site A, and then goes to the exit:

```
(goto_ctrlRoom_t0)
(goto_siteA_t1)
(goto_exit_t2)
```

As the robot executes each action, his epistemic state changes. From here, the values of initial modal sentences can change based on the actions the robot takes: once the robot visits the control room, he learns the reactor type from the indicator light. If location A is visited and the reactor is of type breeder, then the robot will know whether there is a radiation leak at A and $KWradA$ becomes true. Likewise, if location B is visited and the reactor is of type pebble, then $KWradB$ becomes true. On the other hand, if the robot visits location A and the reactor type is pebble, no information is gained.

5.2.3 Goal

The goal of the agent is to determine whether there is a radiation leak in the facility (at either site A or B, depending on the reactor type). We assert this at time-point 3, since the observer assumes the goal has been is achieved when the agent exits the facility (after 3 actions, in this scenario). Knowing the goal of the robot plays a critical role in interpreting the actions of the robot.

```
(or (KW:t3 radA) (KW:t3 radB))
```

Importantly, the goal is expressed using KW operators, because the observer must be ambivalent about the outcome (radiation levels) in this domain. The goal of the robot is simply to check whether there is radiation or not, and by the end of the run, the observer knows the robot should know whether this is true or not.

By explicitly stating the goal of the robot in the knowledge base in this way, we satisfy both of the previous assumptions: this single statement elegantly asserts (1) that the goal is reached while simultaneously (2) expressing the goal in terms agnostic to the true value of the radiation.

5.2.4 Frame Axioms

Clearly, the robot does not have the power of omnipresence, but rather it must carry out actions sequentially one after the other – this is yet another assumption that must be explicitly defined within the knowledge base.

Propositions representing the robot's movement (or other fluents) need to be extended with time indices, while the remaining propositions do not change with time (i.e *radA* or *pebble*) as they are immutable facts of the world (i.e. not fluents, but rather literals). A rather unique aspect of the reactor problem is that the robot's state of belief regarding propositions must change over time in order to transition from the initial state to the goal of $KW\ radA \vee KW\ radB$.

The addition of time indices allows the knowledge base to more accurately represent the example, however it also creates a new problem. Consider the scenario in which the robot executes $goto\_CtrlRoom\_t1$ and subsequently learns the reactor type. Beginning in the next time, we can now assert this as $KW{:}t2\ pebble$. Say the robot now executes $gotoA\_t2$. Is $KW{:}t3\ pebble$ true for the robot? Unfortunately, the answer is no; the robot cannot prove that $KW{:}t3\ pebble$ is true, nor can it show that $KW{:}t3\ pebble$ is false – such information is lost in the transition from time step 2 to 3 (because of the independent semantics of the operators). This is an example of the frame problem, as we have not explicitly described

what propositions remain unchanged as an action is taken nor have we explicitly allowed for knowledge to persist over time.

Frame axioms are used to encode such knowledge about the preconditions and effects of actions in this domain. These are based on Situation Calculus axioms that have been propositionalized (since EpiRes is built on a modal-propositional language). Hence, for an action at time-point i, the pre-conditions are assertions (antecedents) that must be satisfied at time-point i, and the effects are consequents that would be true in the successor state, $i+1$ (note that time index of the action is associated with the predecessor state, i; the effects appear in the successor state/next time-point, $i+1$). The time-points appended as necessary to propositions, actions, or operators. Then, T copies of each rule would be needed, one for each time step $1 \ldots T$, as is typical with propositionalizations (using a finite set of constants to replace a first-order variable for situations similar to SatPlan).

For example, to encode a rule that, if the agent opens the door to room R1, which requires being at that location as a pre-condition, then the door will be open in the successor state:

```
(-> (and (at_R1_t0) (do_open_doorR1_t0))
       (doorR1_open_t1))
(-> (and (at_R1_t1) (do_open_doorR1_t1))
       (doorR1_open_t2))
...
```

In the Reactor domain, the important effects to capture are knowledge-level effects. If the robot enters the control room, it will know the reactor type (*pebble* means type=pebble; ¬*pebble* means type=breeder). For simplicity, this action has no pre-conditions.

```
(-> (goto_ctrlRoom_t0) (KW:t1 pebble))
(-> (goto_ctrlRoom_t1) (KW:t2 pebble))
...
```

If the robot enters site A and the reactor is of type breeder ($\neg pebble$), then the agent will be able to sense the radiation level and know whether $radA$ is true.

```
(-> (and (goto_siteA_t0) (not pebble)) (KW:t1 radA))
(-> (and (goto_siteA_t1) (not pebble)) (KW:t2 radA))
...
```

Conversely, if the reactor is pebble-type and the agent visits site B, it will know whether there is radiation at B in the subsequent time-step:

```
(-> (and (goto_siteB_t0) pebble) (KW:t1 radB))
(-> (and (goto_siteB_t1) pebble) (KW:t2 radB))
...
```

It is important to use the agnostic KW operators to describe the effects of these actions because, from the observer's point of view, the observer will know that the robot will learn/acquire this information (e.g. $pebble$ or $radA$), without the observer necessarily knowing the truth-values of these propositions itself.

To carry out automated deduction using resolution refutation, however, we need to re-formulate these rules as Frame Axioms, to encode all the ways in which facts could be true in a given situation, encompassing both direct effects of actions, and being true in the previous time-step and being unaffected by other actions. In the case of something like knowing whether there is radiation at site A, this could be satisfied by the conditions above - the robot visits that site and the reactor is not pebble-type - or alternatively, the robot could have known whether there is radiation at site A in the previous time step and simply remembered, since no other action would negate this. We write this exhaustive list as a biconditional (again, copied for each time-step, and with analogous frame axioms for site B, where radiation can be sensed for pebble reactors):

```
(<-> (KW:t1 radA)
 (or (KW:t0 radA) (and goto_siteA_t0 (not pebble))))
(<-> (KW:t2 radA)
 (or (KW:t1 radA) (and goto_siteA_t1 (not pebble))))
...
(<-> (KW:t1 radB)
 (or (KW:t0 radB) (and goto_siteB_t0 pebble)))
(<-> (KW:t2 radB)
 (or (KW:t1 radB) (and goto_siteB_t1 pebble)))
...
```

This solves the frame problem by properly describing knowledge transfer between time steps as sets of axioms focused on the fluents themselves.

5.2.5 Mutual Exclusion

Finally, for completeness, the KB requires adding mutual exclusion axioms among actions at each time point to enforce the fact that the robot can only execute one action at a time (hence the other action propositions are implied to be false). This is needed in some of the proofs below to infer, for example, that site B was *not* visited in scenario A, since only the selected actions in the plan were specified above. There would be copies of similar rules for each action and time-point, excluding all other actions.

```
(-> goto_ctrlRoom_t0 (and (not goto_siteA_t0)
      (not goto_siteB_t0) (not goto_Exit_t0)))
...
```

With all of this information now explained, we can turn our attention to the complete knowledge base for the reactor problem. In summary, the knowledge base for the reactor scenario is encoded in a format that describes the actions and epistemic state of the robot across discrete time steps in a SATplan style as to avoid the frame problem. Sentences

included describe the initial state of the world, the robot's goal, the robot's actions, mutual exclusion laws, and several axioms describing the particular reactor details and knowledge transfer between time steps.

5.3 Reactor A Scenario

In this scenario, the observer watches the robot go to the control room, then visits site A, then goes to the exit. We assert these actions:

```
goto_ctrlRoom_t0, goto_siteA_t1, goto_Exit_t2
```

The knowledge base has 36 modal-propositional sentences written with the KW operator which is appended in full in Table A.1 in the Appendix. After applying the KW transformation and converting the sentences into CNF, we are left with 378 modal propositional clauses for EpiRes to work with. Under our previous assumptions (the robot achieves its goal and that results are agnostic to the radiation reading), the observer should be able to infer that the reactor must be of type breeder (that is, $\neg pebble$). As described in Section 4.1.2, we negate the query and add it to the KB and run EpiRes in search of the empty clause.

By asserting *pebble*, EpiRes finds the empty clause in 5,312 resolutions generating a proof tree with approximately 424 clauses (after factoring and filtering out repeated clauses and contradictory clauses that can resolve with themselves)[1].While the proof tree is too larger to display in this paper, the outline of the proof is as follows:

- The robot knows whether $radA$ or $radB$ since we assert that it reached its goal.
- The robot does not know whether $radB$ in $t0$ and since he did not visit location B in $t0$, he will not know it in $t1$ (by our frame axioms). This repeats in each time step from $t0, \ldots, t3$. Note that we know the robot did not visit location B via the mutual exclusion axioms.

[1]The values for number of resolutions and clauses is an aggregate of several sub-proofs generated by EpiRes. Due to current limitations of efficiency, this proof had to be carried out piece-wise by proving intermediate conclusions to support the overall proof. For more details, reference the appendix Table A.2.

- We resolve not knowing whether $radB$ in $t3$ with the goal to conclude the robot must have achieved his goal by knowing whether $radA$ in $t3$.
- There are two ways for the robot to know whether $radA$ in $t3$: either (i) he knew it to begin with or (ii) he visited location A while $\neg pebble$ is true. This rule is defined by the a frame axiom.
- The robot started by not knowing whether $radA$ in $t0$ thus it cannot be case (i). Additionally, since we asset $pebble$, the robot would never be able to know whether $radA$ in any time step, thus case (ii) is also false.
- Hence, by the frame axioms, the robot does not know whether $radA$ in $t3$. This resolves with the fact that the robot must have achieved his goal by knowing whether $radA$ in $t3$ thus leaving us with a contradiction.
- Therefore, the observer can conclude that the original query of $\neg pebble$ is entailed.

5.4 Reactor B Scenario

In this scenario, the observer watches the robot go to the control room, then visits site B, then goes to the exit. This is the same case as before replacing site A with site B. We assert these actions:

```
goto_ctrlRoom_t0, goto_siteB_t1, goto_Exit_t2
```

The knowledge base is constructed in the same fashion as in scenario A, only with the robot's actions changed. After such actions, the observer should be able to infer that the reactor must be of type pebble. This follows a symmetric line of thinking explained above for the reactor A scenario.

5.5 Reactor AB Scenario

In this scenario, the observer watches the robot go to the control room, then visits site A, visits site B, and then goes to the exit. We assert these actions:

`goto_ctrlRoom_t0, goto_siteA_t1, goto_siteB_t2, goto_Exit_t3`

We again maintain the same knowledge base used in the previous scenarios with an amended list of actions of the robot. After such a sequence of actions, the observer should not be able to infer *pebble* nor $\neg$*pebble*. In testing each query, EpiRes failed to find the empty clause after 100,000 iterations at which we terminated the program. However, this is not a true proof as resolution refutation theorem proving systems (as a whole) are unable to prove that a query is not entailed.

In order to circumvent such a problem, the query could be restructured to prove that the observer believes that *pebble* is possible and that $\neg$*pebble* is possible. Since the current KB assumes that every sentence is implicitly nested within the context of $\Box_{obs}$, an attempt to prove such a query would require an explicit addition of the modal context as well as some nuanced rules drawing relationships between the knowledge state of the observer and the robot. Unfortunately, due to time restrictions, these changes were not made and will be left as a part of the extension of this research in the future.

Finally, we note that despite the fact that this proof is difficult for EpiRes, it can be simply reasoned about. The key is that the observer does not know how the robot accomplished his goal. Since the robot has the goal to KW: $radA$ or KW: $radB$ and has previously only went to a single site (A or B), it is easy to determine the reactor type. Now, for the case where the robot goes to both locations, there is one location where the goal is reached and another location where no information is gained - the observer is unable to infer which location is which thus leaving him unable to determine the reactor type. Modeling this scenario is a substantial task left for future work.

6. DISCUSSION

In summary, we have successfully defined an algorithm for resolving m-clauses written in the KD modal language that was implemented in a full resolution-refutation inference system that can be used for epistemic reasoning (EpiRes). Moreover, we demonstrated that EpiRes works as expected for non-trivial tests within two application domains. Additionally, we introduced the KW operator which significantly simplified the modal clauses particularly for the reactor domain. Despite these advancements, we also observed limitations (especially around efficiency), that pose challenges for future development. This section will conclude the paper as we discuss continued work and these limitations further as well as a brief section that compares EpiRes to similar theorem proving systems.

6.1 Continued Work

There are several opportunities to continue developing EpiRes. Most importantly, we want to rigorously prove that EpiRes is a valid automated theorem prover for the modal logic of KD. Since it has previously been shown that the modal resolution rules (sigma, gamma, implication) are correct and complete [Enjalbert and del Cerro1989], all that remains to show is that EpiRes is a faithful implementation of such rules.

Specifically, we are interested in showing that EpiRes (1) implements all Σ and Γ rules properly, (2) uses a correct interpretation of matching modal contexts and (3) is complete when specifically using linear resolution in a modal logic. Proofs of these properties are on-going work.

An additional continuation of EpiRes would include extending functionality to support the KD45 language. KD45 can be understood as a union of axioms where $KD45 = \{KD \cup 4 \cup 5\}$ such that

$$4\text{: } \Box\phi \rightarrow \Box\Box\phi$$

$$5\text{: } \neg\Box\phi \rightarrow \Box\neg\Box\phi$$

Axioms 4 and 5 encode positive and negative introspection respectively and when combined with KD to form KD45, is a reasonable approximation of representing beliefs [Hintikka1962]. By adding these addition axioms, EpiRes would have the ability to represent more complex epistemic reasoning chains via abstraction to a higher or lower modal context. Luckily, this would not be difficult to add to EpiRes given the current representation of modal propositional clauses as hierarchical trees. Whereas matching modal contexts currently compare literals at the same level, a type of trailing pointer can be added to also consider making comparisons with a modal context and one level higher (such as $[\vee, \Box, \vee, \Box]$ and $[\vee, \Box]$).

6.2 Limitations

Although resolution-refutation based theorem-proving systems are popular, they do have an important implication to consider. Resolution (of any sort) is unable to prove that a query is *not* entailed. In other words, resolution shows that something is unsatisfiable by reaching the empty clause, but it cannot show that something is satisfiable.

The only way to show that a query is satisfiable is to use a larger argument to prove that it is impossible to service the empty clauses. In such cases, resolution-refutation based theorem-proving systems, including EpiRes, run continuously generating clauses without halting[1]. It is unrealistic to generate all possible clauses in the context of the modal logic used in this paper.

Another limitation is the current optimization state of EpiRes. Unfortunately, several of the more complex queries tested in the reactor domain were not able to reach the empty clause independently. In order to aid EpiRes in reaching such conclusions, intermediate queries were tested, proved, and then asserted as a boost to prove the more complex query. For example, in the reactor domain it is necessary to transfer knowledge across time steps. A natural selection for intermediate queries in this scenario is choosing facts to prove in earlier time steps. One avenue is improving the heuristic used for selecting resolvable pairs

[1]We had a typical limit of 100,000 iterations per query before halting.

of clauses. Properly sorting clauses has a big impact on the order of resolutions and hence the number of iterations before finding the empty clause. This is another area of on-going work.

Finally, we acknowledge the limitation of working on propositional logic. Despite the fact that propositional logics grant us desired language attributes such as declarative, compositional, context-independent, and unambiguous, it is not nearly as powerful as first-order logic. By focusing on relations that may or may not hold, first-order logic is more expressive and concise while representing sentences that more closely align with natural thought. Although resolution has been lifted to first order logic [Gilmore1960], it is significantly more complicated with respect to a modal-first-order logic. We save this extension for a separate project as constructing EpiRes for a modal-propositional logic is the first step towards future systems.

6.3 Comparison to Similar Systems

Although EpiRes is not the first of its kind, there are major differences between it and existing systems that warrant benefits of using our construction. From a high level, one of the most similar systems to EpiRes's implementation of ZIPPER is the Recursive Resolution algorithm written as RR [Chan1987]. Although both consider recursive algorithms to resolve modal clauses, the construction of the modal clause representation drastically differs and directly affects the complexity of the resolution process. The RR method applies a form of Skolemization to $\Diamond$ operators and then uses a complex process (77 lines of LISP) to determine if two modal clauses are resolvable. On the other hand, our deliberate structure of modal clauses as nested ASTs allow EpiRes to easily check if modal contexts match below applying the resolution rules. In this way, EpiRes can be seen as a tail recursive implementation of a term rewriting system.

Related to automated theorem provers, we also consider model checking or tableaux methods leveraged to make inferences within modal languages [Gasquet *et al.*2005]. Tableaux methods take the approach of taking logical sentences and breaking them down into their

elementary components which are either left open or closed off if a contradiction is found. These semantic model checking works well, however does not scale to support the size that syntactic proof procedures like EpiRes are able to handle.

Finally, we briefly comment on the difference between our introduction of the KW operator to the $\mathcal{VSK}$ logic [Wooldridge and Lomuscio2000]. Within $\mathcal{VSK}$, the operators most similar to KW are $\mathcal{V}$ and $\mathcal{S}$. $\mathcal{V}\phi$ describes the visibility of ϕ which we can understand this to mean whether or not something is able to be sensed (i.e $\mathcal{V}\phi \not\Longrightarrow \mathcal{S}\phi$ however $\mathcal{S}\phi \Longrightarrow \mathcal{V}\phi$). Thus in the reactor domain, we consider $light \leftrightarrow pebble \leftrightarrow \mathcal{V}radB \leftrightarrow \neg\mathcal{V}radA$.

The $\mathcal{S}$ operator describes if something is able to be sensed. Moreover, $\mathcal{S}\phi$ means the agent senses that ϕ is true (and that ϕ actually is true) and similarly for $\mathcal{S}\neg\phi$. In contrast, KW is designed to be ambivalent. The beauty of $KW\phi$ is that it does not commit that ϕ is true or false, but rather makes a statement of the belief state of a particular literal.

REFERENCES

[Baltag *et al.*1998] A. Baltag, L.S. Moss, and S. Solecki. The logic of public announcements, common knowledge, and private suspicions. In I. Gilboa, editor, *Conference on Theoretical Aspects of Rationality and Knowledge*, pages 43–56, 1998.

[Chan1987] M.-C. Chan. The Recursive Resolution Method for Modal Logic. *New Generation Computing*, pages 155–183, 1987.

[Cohen and Levesque1990] P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(2–3):213–261, 1990.

[Dixon *et al.*1998] C. Dixon, M. Fisher, and M. Wooldridge. Resolution for Temporal Logics of Knowledge. *Journal of Logic & Computation*, pages 345–372, 1998.

[Enjalbert and del Cerro1989] P. Enjalbert and L.F. del Cerro. Modal resolution in clausal form. *Theoretical Computer Science*, 65:1–33, 1989.

[Fagin *et al.*1995] R. Fagin, J. Y. Halpern, Y. Moses, and M. Vardi. *Reasoning About Knowledge*. MIT Press, Cambridge, Massachusetts, 1995.

[Gasquet *et al.*2005] O. Gasquet, A. Herzig, D. Longin, and M. Sahade. LoTREC: Logical Tableaux Research Engineering Companion. In B. Beckert, editor, *Automated Reasoning with Analytic Tableaux and Related Methods TABLEAUX 2005. Lecture Notes in Computer Science*, volume 3702. Springer, Berlin, Heidelberg, 2005.

[Gilmore1960] Paul C. Gilmore. A proof method for quantification theory: Its justification and realization. *IBM J. Res. Dev.*, 4:28–35, 1960.

[Halpern and Moses1992] J. Halpern and R. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54:319–379, 1992.

[Hintikka1962] J. Hintikka. *Knowledge and Belief*. Cornell University Press, 1962.

[Kominis and Geffner2015] F. Kominis and H. Geffner. Beliefs in multiagent planning: From one agent to many. In *Proceedings of the Twenty-fifth International Conference on Automated Planning and Scheduling*, pages 147–155, 2015.

[Loveland1970] D. W. Loveland. A linear format for resolution. In *Proc. IRIA Symp. Automatic Demonstration*, pages 147–162, Versailles, France, 1970.

[Luckham1970] D. Luckham. Refinements in resolution theory. In *Proc. IRIA Symp. Automatic Demonstration*, pages 163–190, Versailles, France, 1970.

[Rao1996] A. S. Rao. Decision procedures for propositional linear-time belief-desire-intention logics. In *Intelligent Agents II: Agent Theories, Architectures, and Languages*, page 33–48. Springer, Berlin, Heidelberg, 1996.

[Sebastiani and Vescovi2009] R. Sebastiani and M. Vescovi. Automated Reasoning in Modal and Description Logics via SAT Encoding, the Case Study of K(m)/ALC-Satisfiability. *Journal of AI Research*, 35:343–389, 2009.

[van Benthem *et al.*2018] Johan van Benthem, Jan van Eijck, Malvin Gattinger, and Kaile Su. Symbolic model checking for Dynamic Epistemic Logic—S5 and beyond. *Journal of Logic and Computation*, 28(2):367–402, 2018.

[Wooldridge and Lomuscio2000] M. Wooldridge and A. Lomuscio. Multi-agent VSK Logic. In *Logics In Artifical Intelligence*, pages 300–312, 2000.

[Zhang *et al.*2018] Y. Zhang, D.A. Shell, and J.M. O'Kane. Finding plans subject to stipulations on what information they divulge. In *Proceedings of International Workshop on the Algorithmic Foundations of Robotics*, 2018.

APPENDIX A: TABLES

Table A.1: Reactor A Scenario

Sentence	**Annotation**
(or atStart_t0)	Initial
(or (not (KW:t0 (or radA))))	Initial
(or (not (KW:t0 (or radB))))	Initial
(or (not (KW:t0 (or pebble))))	Initial
(or (KW:t3 (or radA)) (KW:t3 (or radB)))	Goal
(or gotoCrtlRoom_t0)	Action
(or gotoA_t1)	Action
(or gotoExit_t2)	Action
(<-> (KW:t1 (or pebble)) (or (KW:t0 (or pebble)) gotoCrtlRoom_t0))	Axiom
(<-> (KW:t2 (or pebble)) (or (KW:t1 (or pebble)) gotoCrtlRoom_t1))	Axiom
(<-> (KW:t3 (or pebble)) (or (KW:t2 (or pebble)) gotoCrtlRoom_t2))	Axiom
(<-> (KW:t4 (or pebble)) (or (KW:t3 (or pebble)) gotoCrtlRoom_t3))	Axiom
(<-> (KW:t1 (or radA)) (or (KW:t0 (or radA)) (and gotoA_t0 (not pebble))))	Axiom
(<-> (KW:t2 (or radA)) (or (KW:t1 (or radA)) (and gotoA_t1 (not pebble))))	Axiom
(<-> (KW:t3 (or radA)) (or (KW:t2 (or radA)) (and gotoA_t2 (not pebble))))	Axiom
(<-> (KW:t4 (or radA)) (or (KW:t3 (or radA)) (and gotoA_t3 (not pebble))))	Axiom
(<-> (KW:t1 (or radB)) (or (KW:t0 (or radB)) (and gotoB_t0 pebble)))	Axiom
(<-> (KW:t2 (or radB)) (or (KW:t1 (or radB)) (and gotoB_t1 pebble)))	Axiom
(<-> (KW:t3 (or radB)) (or (KW:t2 (or radB)) (and gotoB_t2 pebble)))	Axiom
(<-> (KW:t4 (or radB)) (or (KW:t3 (or radB)) (and gotoB_t3 pebble)))	Axiom
(-> gotoA_t0 (and (not gotoB_t0) (not gotoCrtlRoom_t0) (not gotoExit_t0)))	Mutual exclusion
(-> gotoB_t0 (and (not gotoA_t0) (not gotoCrtlRoom_t0) (not gotoExit_t0)))	Mutual exclusion
(-> gotoCrtlRoom_t0 (and (not gotoB_t0) (not gotoA_t0) (not gotoExit_t0)))	Mutual exclusion
(-> gotoExit_t0 (and (not gotoB_t0) (not gotoCrtlRoom_t0) (not gotoA_t0)))	Mutual exclusion
(-> gotoA_t1 (and (not gotoB_t1) (not gotoCrtlRoom_t1) (not gotoExit_t1)))	Mutual exclusion
(-> gotoB_t1 (and (not gotoA_t1) (not gotoCrtlRoom_t1) (not gotoExit_t1)))	Mutual exclusion
(-> gotoCrtlRoom_t1 (and (not gotoB_t1) (not gotoA_t1) (not gotoExit_t1)))	Mutual exclusion
(-> gotoExit_t1 (and (not gotoB_t1) (not gotoCrtlRoom_t1) (not gotoA_t1)))	Mutual exclusion
(-> gotoA_t2 (and (not gotoB_t2) (not gotoCrtlRoom_t2) (not gotoExit_t2)))	Mutual exclusion
(-> gotoB_t2 (and (not gotoA_t2) (not gotoCrtlRoom_t2) (not gotoExit_t2)))	Mutual exclusion
(-> gotoCrtlRoom_t2 (and (not gotoB_t2) (not gotoA_t2) (not gotoExit_t2)))	Mutual exclusion
(-> gotoExit_t2 (and (not gotoB_t2) (not gotoCrtlRoom_t2) (not gotoA_t2)))	Mutual exclusion
(-> gotoA_t3 (and (not gotoB_t3) (not gotoCrtlRoom_t3) (not gotoExit_t3)))	Mutual exclusion
(-> gotoB_t3 (and (not gotoA_t3) (not gotoCrtlRoom_t3) (not gotoExit_t3)))	Mutual exclusion
(-> gotoCrtlRoom_t3 (and (not gotoB_t3) (not gotoA_t3) (not gotoExit_t3)))	Mutual exclusion
(-> gotoExit_t3 (and (not gotoB_t3) (not gotoCrtlRoom_t3) (not gotoA_t3)))	Mutual exclusion

Table A.2: Reactor A Intermediate Proofs

Number	**Sentence**	**Iterations**	**Nodes in Proof**
1	`(or (not (KW:t2 radA)) (KW:t1 radA))`	23	31
2	`(or (not (KW:t3 radB)))`	2392	131
3 (given 1)	`(or (not (KW:t3 radA)))`	2835	137
4 (given 2,3)	`(or )`	62	125

As described in this paper, the overall structure of the proof for the reactor A scenario is that given *pebble*, EpiRes proves `(KW:t3 radA))` and `(not (KW:t3 radA))` to reach a contradiction ultimately proving $\neg$*pebble*. Breaking this proof down, Table A.2 describes which intermediate steps were individually proved and assert to assist in the greater proof. Combining these number, we reach the aggregate numbers presented previously.