Modal Logic S5 in Answer Set Programming with Lazy Creation of Worlds

Mario Alviano[1], Sotiris Batsakis[2,3], and George Baryannis[3]

[1] University of Calabria, Rende, Italy,
alviano@mat.unical.it,
[2] Technical University of Crete, Chania, Crete, Greece
sbatsakis@isc.tuc.gr,
[3] School of Computing and Engineering, University of Huddersfield, Huddersfield, UK
g.bargiannis@hud.ac.uk

Abstract. Modal logic S5 is used extensively for representing knowledge that includes statements about necessity and possibility, owing to its simplicity in handling chained modal operators. Significant research effort has been devoted in developing efficient reasoning mechanisms over complex S5 formulas, resulting in various solvers taking advantage of the boolean satisfiability problem (SAT). Among them, the most performant solver implements a heuristic for identifying worlds that can be merged, hence reducing the size of SAT instances to be checked. Recently, Answer Set Programming (ASP) has also been considered, and different ASP encodings were proposed and tested, reaching state-of-the-art performance. In particular, a heuristic for identifying the propositional atoms that are relevant in every world resulted in a performance gain in previous experiments. This work addresses the open question of whether the aforementioned two heuristics can be combined, as well as possibly enabling lazy instantiation of the resulting encodings, and what their potential impact is on the performance of the ASP-based solver. Experiments show that lazy creation of worlds provides some further performance gain to the ASP-based solver on the tested instances.

Keywords: Modal Logic, S5, Answer Set Programming

1 Introduction

Modal logics extend standard logic-based languages to include the ability to express modalities qualifying truth statements, and have numerous applications, among them legal reasoning [4], linguistics [15] and multi-agent systems [13]. S5 is one of the most well-known and studied syntax systems, while Kripke semantics [11] is the commonly accepted model-theoretic approach to defining modal logic semantics. The defining characteristic of S5 is that it allows simplifying complex sequences of modal operators by retaining only the last one in the sequence and pruning all the rest. Kripke semantics interprets formulas as true or false in a set of possible worlds and defines an accessibility relation that can link pairs of these worlds, meaning that if a formula is true in one world then it is possibly true in all other worlds that lead to it. While, in general, the accessibility relation can be any binary relation, each modal logic syntax system restricts

its definition. In the case of S5, the accessibility relation is an equivalence relation, satisfying all three of the reflexive, symmetric and transitive properties.

S5 satisfiability is an NP-complete problem [12], and it has been addressed by adopting techniques like resolution [16], tableau [9] and propositional satisfiability (SAT). Solvers relying on SAT have shown great potential, but their need for Skolemisation to represent truth values in different possible worlds may lead to long formulas. State-of-the-art solvers use different methods to attack such an issue, with S52SAT [6] reducing the number of possible worlds that need to be explored to find a model and using structural caching techniques, and S5CHEETAH [10] using formula normalisation and optimising conflicts between modalised literals through the use of graphs.

Recently, a novel S5 solver named S5PY [1] was designed and implemented based on Answer Set Programming (ASP) [14,5]. ASP was considered due to its close relationship with SAT and the readability and configurability afforded by ASP encodings due to their logic programming nature [3,2]. The most efficient algorithm implemented by S5PY is based on a heuristic for identifying the propositional atoms that are relevant in every world. Whether such a heuristic can be combined with the heuristic employed by S5CHEETAH to merge non-conflictual worlds remains an open question, which is addressed in this work. Additionally, we report on a lazy algorithm for the incremental instantiation of the ASP encoding used by S5PY, where worlds associated with modalised literals are not immediately introduced but delayed until required.

The main contributions of this paper can be summarised as follows:

- The heuristic implemented by S5CHEETAH using graph colouring to identify non-conflictual worlds is integrated in S5PY. The main differences between the way such a heuristic is implemented in S5CHEETAH and S5PY are discussed in Section 6.
- The most efficient encoding of S5PY presented in [1] is redesigned so that lazy instantiation can be enabled, as well as the heuristic using graph colouring. In this way, the underlying ASP system is not obliged to eagerly materialise all propositional atoms and rules before starting the search for an answer set, and in fact it may complete its computational task without materialising worlds associated with false modalised literals.
- An empirical evaluation of four different configurations of S5PY is reported, showing that some small performance gain is achieved by lazy creation of worlds, while merging worlds introduces overhead to the implemented solver.

2 Background

S5 extends propositional logic with the modal operators $\Box$ for encoding *necessity* and $\Diamond$ for encoding *possibility*. The language is defined by the grammar

$$\phi := p \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \Box\phi \mid \Diamond\phi \tag{1}$$

where p is a *propositional atom* among those of a fixed countably infinite set $\mathscr{A}$. Moreover, logical connectives for implication and equivalence are used as syntactic sugar with the usual meaning, i.e. $\phi \rightarrow \psi := \neg\phi \vee \psi$ and $\phi \leftrightarrow \psi := (\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi)$, for

every pair of formulas ϕ and ψ. The *complement* of a propositional literal is defined as usual, i.e. $\overline{p} = \neg p$ and $\overline{\neg p} = p$ for all $p \in \mathscr{A}$, and the notation is naturally extended to sets of propositional literals.

The semantics of S5 formulas is given by *Kripke structures*, that is, non-empty sets of *worlds* connected by an accessibility relation; the accessibility relation can be assumed to be total for S5 formulas [7], so for the purposes of this paper only the set of worlds will be used, and actually worlds will be represented in a list so to associate the i-th $\Diamond$-literal with the world in position i. A *world* is an interpretation of propositional logic, that is, a function I assigning a truth value of either 0 (false) or 1 (true) to every propositional atom in $\mathscr{A}$. Interpretations are usually represented by the sets of propositional atoms that are assigned a value of true.

Let $\mathbf{I}$ be the list $[I_0, \ldots, I_n]$ of worlds, for some $n \geq 0$, and let $0 \leq i \leq n$. The *satisfiability relation* $\models$ for S5 formulas is defined as follows: $(\mathbf{I}, i) \models p$ iff $I_i(p) = 1$; $(\mathbf{I}, i) \models \neg\phi$ iff $(\mathbf{I}, i) \not\models \phi$; $(\mathbf{I}, i) \models \phi \wedge \psi$ iff $(\mathbf{I}, i) \models \phi$ and $(\mathbf{I}, i) \models \psi$; $(\mathbf{I}, i) \models \phi \vee \psi$ iff $(\mathbf{I}, i) \models \phi$ or $(\mathbf{I}, i) \models \psi$; $(\mathbf{I}, i) \models \Box\phi$ iff $(\mathbf{I}, j) \models \phi$ for all $j \in [0..n]$; $(\mathbf{I}, i) \models \Diamond\phi$ iff $(\mathbf{I}, j) \models \phi$ for some $j \in [0..n]$. The *satisfiability problem* associated with S5 is the following: given an S5 formula ϕ, is there a list $\mathbf{I} = [I_0, \ldots, I_n]$ (for some $n \geq 0$) such that $(\mathbf{I}, 0) \models \phi$?

Every S5 formula ψ can be transformed into an equi-satisfiable *S5 normal form (S5-NF)* formula [10,1], defined as follows. A *propositional literal* ℓ is either a propositional atom or its negation. A $\Box$-*literal* has the form $\Box(\ell_1 \vee \cdots \vee \ell_n)$, where $n \geq 1$ and $\ell_1, \ldots, \ell_n$ are propositional literals. A $\Diamond$-*literal* has the form $\Diamond(\ell_1 \wedge \cdots \wedge \ell_n)$, where $n \geq 1$ and $\ell_1, \ldots, \ell_n$ are propositional literals. An *S5-literal* is a propositional literal, a $\Box$-literal, or a $\Diamond$-literal. A disjunction of S5-literals is called an *S5-clause*. A formula ϕ is in S5-NF if ϕ is a conjunction of S5-clauses. Let $atoms(\phi)$ and $lits(\phi)$ denote the sets of propositional atoms and literals occurring in ϕ, respectively.

3 S5 Satisfiability Checking Encodings

S5 satisfiability can be expressed in monadic first-order logic, and eventually reduced to SAT by applying Skolemisation and Herbrand expansion, optimising the Tseitin-like transformation by enabling only worlds associated with true $\Diamond$-literals not already witnessed by world 0 [1]. The resulting encoding can be improved by means of some properties related to an overestimate of the literals involved in unit propagation [10,1]. In this section, we first review such encodings by adopting a uniform notation, and then show how they can be combined in a new encoding.

Let ϕ be an S5-NF formula, and let us fix an enumeration $\Box\psi_1^{\Box}, \ldots, \Box\psi_m^{\Box}, \Diamond\psi_1^{\Diamond}, \ldots, \Diamond\psi_n^{\Diamond}$ of its $\Box$- and $\Diamond$-literals, for some $m \geq 0$ and $n \geq 0$. Let $\psi(x)$ denote the monadic first-order formula obtained from ψ by adding argument x to all propositional atoms occurring in ψ. The following propositional atoms are used:

- $p(i)$, representing truth of atom p in world i, for $i \in [0..n]$;
- b_i, representing truth of $\Box\psi_i^{\Box}$, for $i \in [1..m]$;
- d_j, representing truth of $\Diamond\psi_j^{\Diamond}$, for $j \in [1..n]$;
- $implied_j$, representing that $\Diamond\psi_j^{\Diamond}$ is witnessed by world 0, for $j \in [1..n]$.

The *basic encoding* of ϕ, denoted $basic(\phi)$, comprises the following formulas:

1. $b_i \to \psi_i^{\Box}(0)$, and $b_i \wedge d_j \wedge \neg implied_j \to \psi_i^{\Box}(j)$, for all $i \in [1..m]$ and $j \in [1..n]$;
2. $implied_j \leftrightarrow \psi_j^{\Diamond}(0)$, $implied_j \to d_j$, and $d_j \wedge \neg implied_j \to \psi_j^{\Diamond}(j)$ for all $j \in [1..n]$;

where any remaining propositional atom p (ie. those not under the scope of any modal) is replaced by $p(0)$. Intuitively, formulas in the first group enforce that clauses in $\Box$-literals are satisfied in all worlds associated with *true-and-not-implied* $\Diamond$-literals, and formulas in the second group define implied $\Diamond$-literals as those witnessed by world 0 and impose the satisfaction of true-and-not-implied $\Diamond$-literals in their worlds.

Example 1 (Running example). Let ϕ be $\Box(p \vee q) \wedge \Diamond p \wedge \Diamond \neg p \wedge \Diamond \neg q$. Clauses in $basic(\phi)$ encode the following formulas:

$$
\begin{array}{lll}
b_1 \wedge d_1 \wedge d_2 \wedge d_3 & b_1 \to p(0) \vee q(0) & b_1 \wedge d_1 \wedge \neg implied_1 \to p(1) \vee q(1) \\
b_1 \wedge d_2 \wedge \neg implied_2 \to p(2) \vee q(2) & & b_1 \wedge d_3 \wedge \neg implied_3 \to p(3) \vee q(3) \\
implied_1 \leftrightarrow \quad p(0) & implied_1 \to d_1 & d_1 \wedge \neg implied_1 \to \quad p(1) \\
implied_2 \leftrightarrow \neg p(0) & implied_2 \to d_2 & d_2 \wedge \neg implied_2 \to \neg p(2) \\
implied_3 \leftrightarrow \neg q(0) & implied_3 \to d_3 & d_2 \wedge \neg implied_3 \to \neg q(3)
\end{array}
$$

Formula ϕ is satisfied by $\mathbf{I} = [\{p\}, \{p,q\}, \{q\}, \{p\}]$, and the associated model of $basic(\phi)$ is $I = \{b_1, d_1, d_2, d_3, p(0), p(1), q(1), q(2), p(3)\}$. ■

Proposition 1. *For every S5-NF formula ϕ, $basic(\phi)$ is equi-satisfiable to ϕ.*

Proof. By combining Proposition 3.1 and Theorem 3.1 in [1]. □

The basic encoding can be improved thanks to a property that can be checked by computing an overestimate of the literals involved in unit propagation starting from those in $\Diamond$-literals. Formally, for a set L of literals

$$UP(L) := L \cup \bigcup \left\{ lits(\psi_i^{\Box}) \setminus \{\overline{\ell}\} \mid \ell \in L,\ i \in [1..m], \overline{\ell} \in lits(\psi_i^{\Box}) \right\} \tag{2}$$

$$B_j := \left\{ i \in [1..m] \mid \overline{UP \Uparrow lits(\psi_j^{\Diamond})} \cap lits(\psi_i^{\Box}) \neq \emptyset \right\} \tag{3}$$

Intuitively, $UP(L)$ is the set of literals that may be used to satisfy every $\psi_i^{\Box}$ affected by the assignment of L, $UP \Uparrow lits(\psi_j^{\Diamond})$ is the set of literals reached in this way from the literals in $\psi_j^{\Diamond}$, and B_j represents the set of $\Box$-literals involved in this computation. The *reach encoding* of ϕ, denoted $reach(\phi)$, is obtained from $basic(\phi)$ by removing formulas encoding $b_i \wedge d_j \wedge \neg implied_j \to \psi_i^{\Box}(j)$, for all $i \in [1..m]$ and $j \in [1..n]$ such that $i \notin B_j$ (ie. the truth of $\Box\psi_i^{\Box}$ in the world associated with $\Diamond\psi_j^{\Diamond}$ can be witnessed by world 0).

Example 2 (Continuing Example 1). To construct $reach(\phi)$, we have to compute sets B_1, B_2 and B_3, respectively associated with $\Diamond$-literals $\Diamond p$, $\Diamond \neg p$ and $\Diamond \neg q$. Let us first determine the sets of reached literals from $\{p\}$, $\{\neg p\}$ and $\{\neg q\}$ according to (2): $UP(\{p\}) = \{p\}$, as $\overline{p} \notin lits(p \vee q)$; $UP(\{\neg p\}) = \{\neg p, q\}$, as $\overline{\neg p} \in lits(p \vee q)$ and therefore literals in $lits(p \vee q) \setminus \{\overline{\neg p}\} = \{q\}$ are added to $UP(\{\neg p\})$; $UP(\{\neg p, q\}) =$

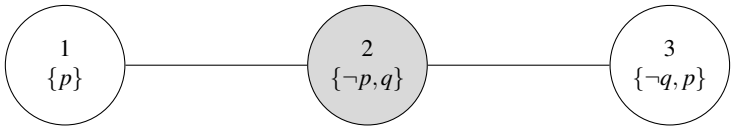


Fig. 1. Diamond conflict graph of formula ϕ from Examples 1–3, with graph colouring

$\{\neg p,q\}$, as $\overline{q} \notin \mathit{lits}(p \vee q)$ and therefore no other literal is added to $UP(\{\neg p,q\})$; $UP(\{\neg q\}) = \{\neg q,p\}$, as $\overline{\neg q} \in \mathit{lits}(p \vee q)$ and therefore literals in $\mathit{lits}(p \vee q) \setminus \{\overline{\neg q}\} = \{p\}$ are added to $UP(\{\neg q\})$; $UP(\{\neg q,p\}) = \{\neg q,p\}$, as $\overline{p} \notin \mathit{lits}(p \vee q)$ and therefore no other literal is added to $UP(\{\neg q,p\})$. Hence, we have that $UP \Uparrow \mathit{lits}(\{p\}) = \{p\}$, $UP \Uparrow \mathit{lits}(\{\neg p\}) = \{\neg p,q\}$ and $UP \Uparrow \mathit{lits}(\{\neg q\}) = \{\neg q,p\}$. Now using (3), $B_1 = \emptyset$ and $B_2 = B_3 = \{1\}$, that is, $\Diamond p$ does not interact with the $\Box$-literal, while $\Diamond\neg p$ and $\Diamond\neg q$ interact with the $\Box$-literal. Accordingly, $\mathit{reach}(\phi)$ is obtained from $\mathit{basic}(\phi)$ by removing clauses encoding $b_1 \wedge d_1 \wedge \neg \mathit{implied}_1 \rightarrow p(1) \vee q(1)$. In fact, such a formula can be satisfied by assigning to $q(1)$ the same truth value of $q(0)$. For example, if I is a model of $\mathit{reach}(\phi)$ such that $I(q(0)) = 1$, then $I \cup \{q(1)\}$ is a model of $\mathit{basic}(\phi)$; similarly, if I is a model of $\mathit{reach}(\phi)$ such that $I(q(0)) = 0$, then $I \setminus \{q(1)\}$ is a model of $\mathit{basic}(\phi)$. ■

Proposition 2. *For every S5-NF formula ϕ, $\mathit{reach}(\phi)$ is equi-satisfiable to $\mathit{basic}(\phi)$.*

Proof. Shown in [1] as Theorem 3.2. □

Alternatively, the basic encoding can be improved by merging some "non-conflictual" worlds, an idea first introduced in [10] and revised here. Formally, the *diamond conflict graph* of ϕ, denoted $\mathscr{G}_\phi$, has vertex set $[1..n]$, and edge set $\{ij \mid \exists \ell \in \mathit{lits}(\psi_i^\Diamond)$ such that $\overline{\ell} \in UP \Uparrow \mathit{lits}(\psi_j^\Diamond)\}$. Let $\mathit{colour} : [1..n] \rightarrow [1..n]$ be a *graph colouring* of $\mathscr{G}_\phi$, that is, colour is such that $\mathit{colour}(i) \neq \mathit{colour}(j)$ holds for all edges ij in $\mathscr{G}_\phi$. The *basic-merge encoding* of ϕ wrt. colour, denoted $\mathit{basic}(\phi, \mathit{colour})$, is obtained from $\mathit{basic}(\phi)$ by replacing

1. $b_i \wedge d_j \wedge \neg \mathit{implied}_j \rightarrow \psi_i^\Box(j)$ with $b_i \wedge d_j \wedge \neg \mathit{implied}_j \rightarrow \psi_i^\Box(\mathit{colour}(j))$, for all $i \in [1..m]$ and $j \in [1..n]$;
2. $d_j \wedge \neg \mathit{implied}_j \rightarrow \psi_j^\Diamond(j)$ with $d_j \wedge \neg \mathit{implied}_j \rightarrow \psi_j^\Diamond(\mathit{colour}(j))$, for all $j \in [1..n]$.

(Note that the basic encoding is a special case of the basic-merge encoding in which colour is the *identity function* $\mathit{id} : i \mapsto i$.) Essentially, the idea underlying the basic-merge encoding is that $\Diamond$-literals whose propositional literals cannot produce any conflict via unit propagation can share the same world.

Example 3 (Continuing Example 2). The diamond conflict graph $\mathscr{G}_\phi$ of ϕ is shown in Figure 1, where vertices are also annotated with the reached literals. The figure also shows the graph colouring $\mathit{colour} = \{1 \mapsto 1, 2 \mapsto 2, 3 \mapsto 1\}$, where colour 1 is white and colour 2 is gray. It turns out that the first and third $\Diamond$-literals can share the same world, and therefore the following formulas of $\mathit{basic}(\phi)$

$$b_1 \wedge d_3 \wedge \neg \mathit{implied}_3 \rightarrow p(3) \vee q(3) \qquad d_2 \wedge \neg \mathit{implied}_3 \rightarrow \neg q(3)$$

are replaced by

$$b_1 \wedge d_3 \wedge \neg \mathit{implied}_3 \rightarrow p(1) \vee q(1) \qquad d_2 \wedge \neg \mathit{implied}_3 \rightarrow \neg q(1)$$

to obtain $\mathit{basic}(\phi, \mathit{colour})$. Note that if I is a model of $\mathit{basic}(\phi)$, then it must satisfy $I(p(1)) = 1$ and $I(q(3)) = 0$ because of the $\Diamond$-literals. Moreover, $I(p(3)) = 1$ because of the $\Box$-literal. Hence, worlds 1 needs p, and worlds 3 needs p and $\neg q$. A model of $\mathit{basic}(\phi, \mathit{colour}$ is then obtained from I by merging worlds 1 and 3 as follows: $(I \cup \{p(1)\}) \setminus \{q(1), p(3), q(3)\}$. ■

Proposition 3. *For every S5-NF formula ϕ and every graph colouring colour of $\mathscr{G}_\phi$, $\mathit{basic}(\phi, \mathit{colour})$ is equi-satisfiable to $\mathit{basic}(\phi)$.*

Proof. Let $I \models \mathit{basic}(\phi, \mathit{colour})$. Hence, the following is a model of $\mathit{basic}(\phi)$:

$$\begin{aligned}\{b_i \in I \mid i \in [1..m]\} \cup \{d_j \in I \mid j \in [1..n]\} \cup \{\mathit{implied}_j \in I \mid j \in [1..n]\} \\ \cup \{p(0) \mid p(0) \in I\} \quad \cup \{p(j) \mid j \in [1..n],\ p(\mathit{colour}(j)) \in I\}.\end{aligned}$$

Essentially, multiple copies of the shared worlds are introduced.

Let $I \models \mathit{basic}(\phi)$. For every colour $i \in [1..n]$, let us define the following set of non-conflictual literals: $L_i := \{\ell \in \mathit{UP} \Uparrow \mathit{lits}(\psi_j^\Diamond) \mid j \in [1..n],\ \mathit{colour}(j) = i,\ I(d_j) = I(\ell) = 1\}$. Hence, the following is a model of $\mathit{basic}(\phi, \mathit{colour})$:

$$\begin{aligned}\{b_i \in I \mid i \in [1..m]\} \cup \{d_j \in I \mid j \in [1..n]\} \cup \{\mathit{implied}_j \in I \mid j \in [1..n]\} \\ \cup \{p(0) \mid p(0) \in I\} \quad \cup \{p(\mathit{colour}(j)) \mid j \in [1..n],\ p \in L_{\mathit{colour}(j)}\}.\end{aligned}$$

Essentially, multiple worlds are merged according to the given graph colouring. □

A new encoding, denoted $\mathit{reach}(\phi, \mathit{colour})$ and called *reach-merge encoding* of ϕ w.r.t. *colour*, can be obtained by combining the ideas presented in [1] and [10]. It comprises all formulas of $\mathit{basic}(\phi, \mathit{colour})$ but those of the form $b_i \wedge d_j \wedge \neg \mathit{implied}_j \rightarrow \psi_i^\Box(\mathit{colour}(j))$, for all $i \in [1..m]$ and $j \in [1..n]$ such that $i \notin B_j$ (i.e. the truth of $\Box\psi_i^\Box$ in the world associated with $\Diamond\psi_j^\Diamond$ can be witnessed by world 0). Also note that the reach encoding is a special case of the reach-merge encoding in which *colour* is the identity function *id*.

Example 4 (Continuing Example 3). The reach-merge encoding of ϕ w.r.t. *colour* is obtained from $\mathit{basic}(\phi, \mathit{colour})$ by removing formula $b_1 \wedge d_3 \wedge \neg \mathit{implied}_3 \rightarrow p(1) \vee q(1)$, similar to how $\mathit{reach}(\phi)$ is obtained from $\mathit{basic}(\phi)$. ■

Lemma 1. *For every S5-NF formula ϕ and every graph colouring colour of $\mathscr{G}_\phi$, $\mathit{reach}(\phi, \mathit{colour})$ is equi-satisfiable to $\mathit{basic}(\phi, \mathit{colour})$.*

Proof. The construction is aligned to the proof of Theorem 3.2 in [1]. $I \models \mathit{basic}(\phi, \mathit{colour})$ implies $I \models \mathit{reach}(\phi, \mathit{colour})$ because $\mathit{reach}(\phi, \mathit{colour}) \subseteq \mathit{basic}(\phi, \mathit{colour})$. As for the other direction, let $I \models \mathit{reach}(\phi, \mathit{colour})$ be such that $I \models b_i \wedge d_j \wedge \neg \mathit{implied}_j \wedge \neg \psi_i^\Box(\mathit{colour}(j))$ for some $i \in [1..m]$ and $j \in [1..n]$ such that $i \notin B_j$ — otherwise $I \models \mathit{basic}(\phi, \mathit{colour})$. Since $b_i \rightarrow \psi_i^\Box(0)$ belongs to $\mathit{reach}(\phi, \mathit{colour})$, we have that $I \models \psi_i^\Box(0)$, and we can

copy a portion of world 0 into world $colour(j)$ to construct a model I' for $reach(\phi, colour)$ such that $I' \models \psi_i^{\Box}(colour(j))$: $L = atoms(\psi_i^{\Box}(colour(j))) \setminus atoms(\psi_j^{\Diamond}(colour(j)))$; $I' = I \setminus \{p(colour(j)) \in L\} \cup \{p(colour(j)) \in L \mid p(0) \in I\}$. By reiterating the process, we end up with a model of $basic(\phi, colour)$. □

Theorem 1. *For every S5-NF formula ϕ and every graph colouring colour of $\mathscr{G}_\phi$, $reach(\phi, colour)$ is equi-satisfiable to ϕ.*

Proof. By combining Lemma 1, Proposition 3 and Proposition 1. □

We conclude this section by observing that both $basic(\phi, colour)$ and $reach(\phi, colour)$ merge worlds, and not $\Diamond$-literals. Actually, the fact that two worlds are merged does not immediately imply that the associated $\Diamond$-literals can be jointly satisfied in that world, but instead that those satisfied in a Kripke structure can also be satisfied in a single world. The following example clarifies this fact.

Example 5. Consider the S5-formula $\phi := \Box p \wedge (\Diamond q \vee \Diamond \neg p)$, which can be satisfied by $\mathbf{I} = [\{p\}, \{p, q\}]$. The two $\Diamond$-literals are non-conflictual, and therefore $colour = \{1 \mapsto 1, 2 \mapsto 1\}$ is a graph colouring of $\mathscr{G}_\phi$. The model of $basic(\phi, colour)$ and $reach(\phi, colour)$ associated with $\mathbf{I}$ is $\{b_1, d_1, p(0), p(1), q(1)\}$, and reflect the fact that the worlds of $\Diamond q$ and $\Diamond \neg p$ can be merged. On the other hand, note that $\Diamond \neg p$ is not satisfied by $\mathbf{I}$, and in fact merging the two $\Diamond$-literals would result into $\Box p \wedge (\Diamond q \wedge \neg p)$, an unsatisfiable formula. ■

4 Implementation in Answer Set Programming

This section presents an ASP implementation of the propositional theory $reach(\phi, colour)$ introduced in the previous section. As already observed, *colour* can also be the identity function *id* (for example, in case one does not want to afford for the computation of a graph colouring). Moreover, many formulas associated with world $colour(j)$ in $reach(\phi, colour)$ are vacuously true if atom d_j is false, that is, if the associated $\Diamond$-literal is assumed false. Based on this observation, an incremental instantiation strategy is introduced, materialising such formulas only after d_j is assigned true for the first time.

The S5-NF formula ϕ and the graph colouring *colour* are encoded by the following facts, denoted $\Pi_{re}(\phi, colour)$ and called the *relational encoding* of ϕ:

- `atom(p)`, for every propositional atom p occurring in ϕ;
- `box(b)`, `pos_box(b,p`$_i$`)`, and `neg_box(b,p`$_j$`)`, for every $\Box$-literal of ϕ of the form $\Box(p_1 \vee \cdots \vee p_m \vee \neg p_{m+1} \vee \cdots \vee \neg p_n)$, with $n \geq 1$ and $n \geq m \geq 0$, and all $i \in [1..m]$ and $j \in [m+1..n]$, where `b` is an identifier for the $\Box$-literal;
- `diamond(d)`, `pos_diamond(d,p`$_i$`)`, and `neg_diamond(d,p`$_j$`)`, for every $\Diamond$-literal of ϕ of the form $\Diamond(p_1 \wedge \cdots \wedge p_m \wedge \neg p_{m+1} \wedge \cdots \wedge \neg p_n)$, with $n \geq 1$ and $n \geq m \geq 0$, and all $i \in [1..m]$ and $j \in [m+1..n]$, where `d` is an identifier for the $\Diamond$-literal;
- `clause(c)`, `pos_clause(c,lit`$_i$`)`, and `neg_clause(c,p`$_j$`)`, for every S5-clause of ϕ of the form $\ell_1 \vee \cdots \vee \ell_m \vee \neg p_{m+1} \vee \cdots \vee \neg p_n$, with $n \geq 1$ and $n \geq m \geq 0$, and all $i \in [1..m]$ and $j \in [m+1..n]$, where `c` is an identifier for the S5-clause and each `lit`$_i$ is the identifier of the associated S5-literal ℓ_i;

- `lrl(p`$_i$`,`s_i`,p`$_j$`,`s_j`)`, where p_i and p_j are propositional atoms, and s_i, s_j belong to $\{$`pos,neg`$\}$, to encode the *reach* relation introduced in Section 3 for literals occurring in some $\Diamond$-literal of ϕ — essentially, set $UP(L)$ in (2);
- `lrb(p`$_j$`,`s_j`,b`$_i$`)`, where p_j is a propositional atom, s_j belongs to $\{$`pos,neg`$\}$, and $i \in [1..m]$, to encode the *reach* relation introduced in Section 3 for $\Box$-literals reached by the associated $\Diamond$-literals — essentially, sets B_j in (3);
- `diamond_world(d,w)`, where `d` is the identifier of a $\Diamond$-literal, and `w` the associated world — essentially, $colour($`d`$) =$ `w`.

Example 6 (Continuing Example 4). Let ϕ be $\Box(p \vee q) \wedge \Diamond p \wedge \Diamond \neg p \wedge \Diamond \neg q$, and *colour* be $\{1 \mapsto 1, 2 \mapsto 2, 3 \mapsto 1\}$. Program $\Pi_{re}(\phi, colour)$ contains the following facts:

```
box(b1).       pos_box(b1,p).      pos_box(b1,q).   atom(p).  atom(q).
diamond(d1).   pos_diamond(d1,p).     clause(c1).   pos_clause(c1,b1).
diamond(d2).   neg_diamond(d1,p).     clause(c2).   pos_clause(c2,d1).
diamond(d3).   neg_diamond(d1,q).     clause(c3).   pos_clause(c2,d2).
                                      clause(c4).   pos_clause(c2,d3).
lrl(p,pos, p,pos).
lrl(p,neg, p,neg).    lrl(p,neg, q,pos).    lrb(p,neg, b1).
lrl(q,neg, q,neg).    lrl(q,neg, p,pos).    lrb(q,neg, b1).
diamond_world(d1,1).  diamond_world(d2,2).  diamond_world(d3,1).
```

Note that replacing *colour* with *id* results in the same set of facts, with the exception of `diamond_world(d3,1)`, which would be replaced by `diamond_world(d3,3)`. ■

In order to possibly enable a progressive instantiation of the ASP program, two sets of rules are used, where the first set is processed once, and the second set is processed at most once for each world associated with some $\Diamond$-literals of ϕ. The following atoms are defined in these sets of rules:

- `true(X)`, to guess true atoms in world 0, true $\Box$-literals, and true $\Diamond$-literals;
- `true(W,X)`, to guess true atoms in each world (different from 0);
- `dra(D,Y)`, to determine atoms reached by $\Diamond$-literals;
- `drb(D,B)`, to determine $\Box$-literals reached by $\Diamond$-literals;
- `world_need_atom(W,X)`, to determine if atoms in a world can be ignored because all associated $\Diamond$-literals are false;
- `active_box_in_world(B,W)`, to determine if a $\Box$-literals is reached by some true-and-not-implied $\Diamond$-literals associated with world `W`.

Let Π_{base} be the following set of rules:

```
r1 :  {true(X)} :- box(X).
r2 :  {true(X)} :- diamond(X).
r3 :  {true(X)} :- atom(X).
r4 :  dra(D,Y) :- pos_diamond(D,X); lrl(X,pos,Y,_).
r5 :  dra(D,Y) :- neg_diamond(D,X); lrl(X,neg,Y,_).
r6 :  drb(D,B) :- pos_diamond(D,X); lrb(X,pos,B).
r7 :  drb(D,B) :- neg_diamond(D,X); lrb(X,neg,B).
r8 :  :- clause(C); not true(X) : pos_clause(C,X);
                      true(X) : neg_clause(C,X).
```

```
r9 :   :- box(B), true(B); not true(X) : pos_box(B,X);
                              true(X) : neg_box(B,X).
r10 :  implied(D) :- diamond(D);     true(X) : pos_diamond(D,X);
                                not true(X) : neg_diamond(D,X).
r11 :  :- diamond(D), implied(D), not true(D).
r12 :  true_not_implied(D) :- diamond(D), true(D), not implied(D).
```

Let $\Pi_w(world)$ be the following set of rules parameterized by constant *world*:

```
r13 :  {true(X,W)} :- W = world; diamond_world(D,W);
             diamond_reach_atom(D,X).
r14 :  :- W = world; diamond(D), true_not_implied(D);
             diamond_world(D,W); pos_diamond(D,X); not true(X,W).
r15 :  :- W = world; diamond(D), true_not_implied(D);
             diamond_world(D,W); neg_diamond(D,X);     true(X,W).
r16 :  world_need_atom(W,X) :- W = world; diamond_world(D,W);
             diamond_reach_atom(D,X); true(D)_not_implied(D).
r17 :  :- W = world; true(X,W); not world_need_atom(W,X).
r18 :  active_box_in_world(B,W) :- W = world;
             diamond_world(D,W), drb(D,B);
             box(B), true(B), diamond(D), true_not_implied(D).
r19 :  :- W = world; active_box_in_world(B,W);
             not true(X,W) : pos_box(B,X); true(X,W) : neg_box(B,X).
```

The above programs can be combined in several ways. Let *colour* be a graph colouring of $\mathscr{G}_\phi$ obtained by a greedy algorithm which considers the vertices in descending order according to their degrees, and assigns to each vertex the smallest available colour in this order [17]. For an ASP program Π, let Search(Π) return a stable model I of Π if any, or otherwise $\bot$. We define the following four algorithms for S5 satisfiability checking:

A1. Search($\Pi_{re}(\phi, id) \cup \Pi_{base} \cup \bigcup_{i\in[1..n]} \Pi_w(i)$) — no merge, no lazy;
A2. Search($\Pi_{re}(\phi, colour) \cup \Pi_{base} \cup \bigcup_{i\in[1..n]} \Pi_w(i)$) — merge, no lazy;
A3. SearchWithLazyWorlds(ϕ, id) — no merge, lazy;
A4. SearchWithLazyWorlds($\phi, colour$) — merge, lazy;

where SearchWithLazyWorlds is shown as Algorithm 1 and is characterised by the incremental instantiation of the rules in $\Pi_w(world)$.

5 Evaluation

The ASP encodings presented in Section 4 have been implemented into the solver S5PY [1]. The solver is written in Python and uses CLINGO version 5.4.0 [8] to search for answer sets. This section reports on an empirical comparison between the algorithms discussed in the previous section. S5PY and pointers to benchmark files are provided at http://www.mat.unical.it/~alviano/LPNMR2022-s5py.zip. The experiments were run on an Intel Xeon 2.4 GHz with 16 GB of memory. Time and memory were limited to 300 seconds and 15 GB, as in [1]. For each instance solved within these limits, we measured both execution time and memory usage.

Algorithm 1: SearchWithLazyWorlds(ψ: S5-NF formula, *colour*: a graph colouring of $\mathscr{G}_\phi$)

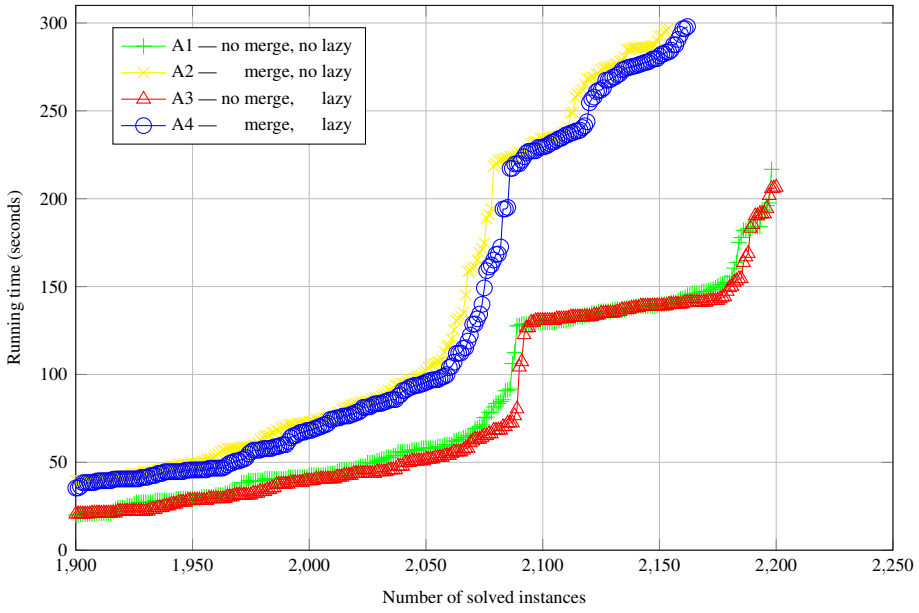
1 $\Pi := \Pi_{re}(\phi, colour) \cup \Pi_{base}$; $W := \emptyset$;
2 **loop**
3 $I := \text{Search}(\Pi)$;
4 **if** $I = \bot$ **then return** $\bot$;
5 $W' := \{w \mid \exists d$ s.t. `diamond_world(`d`,`w`)` and `true(`d`)` are true in $I\} \setminus W$;
6 **if** $W' = \emptyset$ **then return** I;
7 $\Pi := \Pi \cup \bigcup_{world \in W'} \Pi_w(world)$; $W := W \cup W'$;

The first algorithm (A1 — no merge, no lazy) is essentially the reach encoding with no incremental instantiation, the most performant algorithm presented in [1], which is therefore our baseline for comparing how the other techniques discussed in this paper affect the computation of S5PY. As reported in Figure 2, the baseline on the number of solved instances is already over 99%, but a small performance gain is provided by the lazy creation of worlds (2 new solved instances, as well as an improvement of around 0.4 seconds and 14 MiB on average, when A1 is replaced by A3). On the other hand, the heuristic based on the merging of non-conflictual worlds introduces a significant overhead (A2 has 44 timeouts more than A1), which is only minimally compensated by the lazy creation of worlds (A4 solves 8 instances more than A2). Figure 2 also shows a cactus plot of the running time of the tested algorithms, confirming that the lazy creation of worlds provides a minimal but consistent performance gain (A1 vs A3; A2 vs A4).

Figure 3 shows scatter plots comparing the running time of the tested algorithms on each tested instance. A point above the diagonal (dotted red line) means that the algorithm on axis x is faster than the algorithm on axis y. It can be observed that the first two plots (A1 vs A2 and A3 vs A4) evidence the fact that the performance of S5PY deteriorates when non-conflictual worlds are merged. On the other hand, the other two plots (A1 vs A3 and A2 vs A4) witness the small performance gain ascribable to the lazy creation of worlds.

6 Related Work

Directly related work in literature includes the research on the SAT-based S5CHEETAH solver [10] and the original version of the ASP-based solver S5PY [1]. The algorithm that is implemented by S5CHEETAH estimates an upper bound on possible worlds by applying the *graph colourability heuristic*, which is used for identifying non interacting worlds that can be essentially merged. In this work we adapt such an heuristic to S5PY, using the same greedy algorithm for computing graph colouring in polynomial time [17], but differently from S5CHEETAH we associate worlds with $\Diamond$-literals rather than S5-clauses. The argument in favour of associating worlds with S5-clauses is that in this way some $\Diamond$-literals are possibly grouped if they occur under the scope of the same disjunction connective. On the other hand, literals can be combined to form exponentially many clauses, which is the argument in favour of our choice.



Algorithm	Timeouts	Avg. Time (seconds)	Avg. Memory (MiB)
A1 — no merge, no lazy	16	13.2	117
A2 — merge, no lazy	60	20.8	211
A3 — no merge, lazy	14	12.8	103
A4 — merge, lazy	52	20.7	205

Fig. 2. Number of solved instances within a time budget (cactus plot), and overall statistics on 2214 instances (no memory out recorded).

Example 7. Consider the formula $(p \vee \Diamond q) \wedge (\neg p \vee \Diamond q)$. Even if there is only one $\Diamond$-literal, namely $\Diamond q$, there are two different clauses containing some $\Diamond$-literal. In such a case, it is preferable to associate worlds with $\Diamond$-literals, as it is done by S5PY. ■

According to the experiment reported in [1], the original version of the S5PY solver introduced several encodings to address S5 satisfiability checking via ASP, and reached a comparable performance with S5CHEETAH. In particular, the most efficient encoding is the one relying on the reachability relation associated with unit propagation and used to define the reach encoding in Section 3. Such an encoding is now the default strategy used by S5PY, while the other encodings presented in [1] are no longer considered because of their minimal or negative impact on computation. Additionally, the new version of S5PY presented in this work can take advantage of incremental instantiation as implemented by CLINGO.

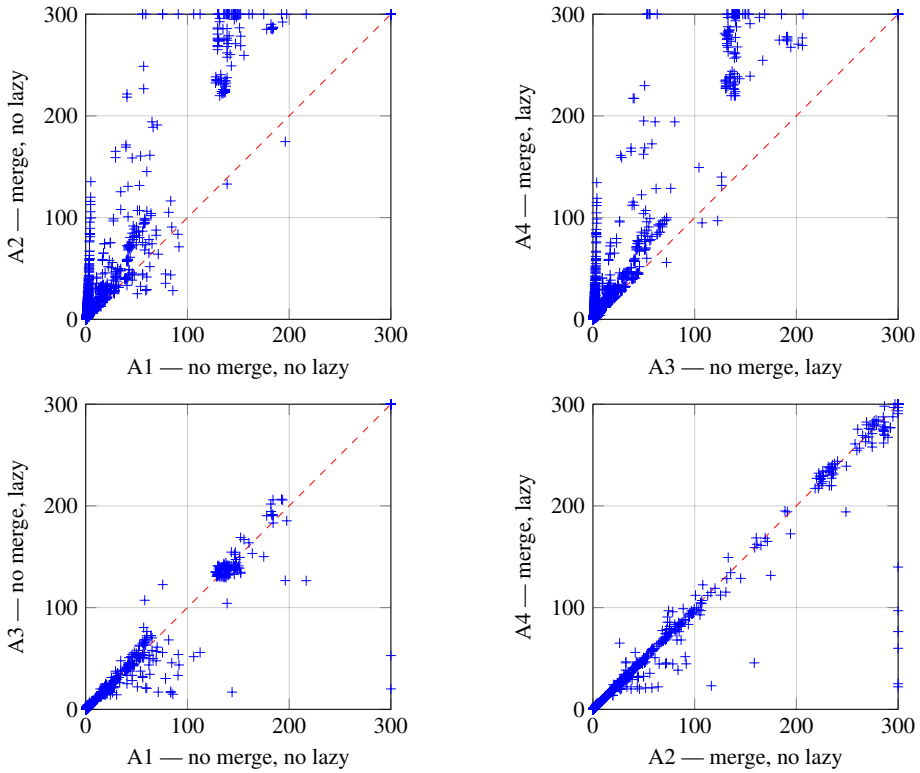


Fig. 3. Instance by instance comparison on execution time (in seconds; timeouts normalised to the limit): Impact of the merging of worlds (top) and the lazy creation of worlds (bottom).

7 Conclusion

This work confirms that ASP is an ideal language for implementing a solver for modal logic S5, and that the good performance achieved by S5PY is mainly ascribable to the reachability relation used to optimise its knowledge compilation algorithm. We have also shown how such a reachability relation can be combined with the graph colourability heuristic implemented by S5CHEETAH, and how the resulting encodings can be incrementally instantiated. Our experiments report no significant gain when combining the two heuristics. This is not necessarily considered a negative result, since the baseline on the number of solved instances is already over 99% and because lazy creation of worlds provides some further performance gain to the original ASP-based solver.

In the future, we intend to further investigate the potential positive impact of incremental instantiation. For instance, we can employ the search heuristic of CLINGO in order to falsify $\Diamond$-literals with the aim of reducing the number of worlds to be actually materialised. We also intend, to explore whether improvements can be made to the graph colourability heuristic to mitigate the overhead it introduces and the increased implementation complexity which negatively affects the performance of S5PY.

References

1. Alviano, M., Batsakis, S., Baryannis, G.: Modal logic s5 satisfiability in answer set programming. Theory and Practice of Logic Programming 21(5), 527–542 (2021)
2. Baryannis, G., Tachmazidis, I., Batsakis, S., Antoniou, G., Alviano, M., Papadakis, E.: A Generalised Approach for Encoding and Reasoning with Qualitative Theories in Answer Set Programming. Theory and Practice of Logic Programming 20(5), 687–702 (2020)
3. Baryannis, G., Tachmazidis, I., Batsakis, S., Antoniou, G., Alviano, M., Sellis, T., Tsai, P.W.: A Trajectory Calculus for Qualitative Spatial Reasoning Using Answer Set Programming. Theory and Practice of Logic Programming 18(3-4), 355—371 (2018), https://doi.org/10.1017/S147106841800011X
4. Batsakis, S., Baryannis, G., Governatori, G., Tachmazidis, I., Antoniou, G.: Legal Representation and Reasoning in Practice: A Critical Comparison. In: Legal Knowledge and Information Systems - JURIX 2018: The Thirty-first Annual Conference. pp. 31–40. IOS Press, Netherlands (2018)
5. Brewka, G., Eiter, T., Truszczyński, M.: Answer set programming at a glance. Communications of the ACM 54(12), 92–103 (2011)
6. Caridroit, T., Lagniez, J.M., Berre, D.L., de Lima, T., Montmirail, V.: A sat-based approach for solving the modal logic s5-satisfiability problem. In: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence. p. 3864–3870. AAAI'17, AAAI Press, Palo Alto, California (2017)
7. Fitting, M.: A simple propositional S5 tableau system. Ann. Pure Appl. Log. 96(1-3), 107–115 (1999), https://doi.org/10.1016/S0168-0072(98)00034-7
8. Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., Wanko, P.: Theory solving made easy with clingo 5. In: Carro, M., King, A. (eds.) Technical Communications of the Thirty-second International Conference on Logic Programming (ICLP'16). Open Access Series in Informatics (OASIcs), vol. 52, pp. 2:1–2:15. Schloss Dagstuhl, Dagstuhl, Germany (2016)
9. Goré, R.: Tableau methods for modal and temporal logics. In: D'Agostino, M., Gabbay, D.M., Hähnle, R., Posegga, J. (eds.) Handbook of Tableau Methods, pp. 297–396. Springer, Netherlands (1999)
10. Huang, P., Liu, M., Wang, P., Zhang, W., Ma, F., Zhang, J.: Solving the satisfiability problem of modal logic S5 guided by graph coloring. In: Kraus, S. (ed.) Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019. pp. 1093–1100. ijcai.org, California (2019)
11. Kripke, S.A.: A completeness theorem in modal logic. The journal of symbolic logic 24(1), 1–14 (1959)
12. Ladner, R.E.: The computational complexity of provability in systems of modal propositional logic. SIAM journal on computing 6(3), 467–480 (1977)
13. Liau, C.J.: Belief, information acquisition, and trust in multi-agent systems—a modal logic formulation. Artificial Intelligence 149(1), 31–60 (2003)
14. Lifschitz, V.: Answer set programming. Springer, Berlin, Heidelberg (2019)
15. Moss, L.S., Tiede, H.J.: 19 applications of modal logic in linguistics. In: Blackburn, P., Van Benthem, J., Wolter, F. (eds.) Handbook of Modal Logic, Studies in Logic and Practical Reasoning, vol. 3, pp. 1031–1076. Elsevier, Amsterdam (2007)
16. Nalon, C., Hustadt, U., Dixon, C.: Ksp: A resolution-based prover for multimodal k, abridged report. In: IJCAI. vol. 17, pp. 4919–4923. ijcai.org, California (2017)
17. Welsh, D.J.A., Powell, M.B.: An upper bound for the chromatic number of a graph and its application to timetabling problems. Comput. J. 10(1), 85–86 (1967), https://doi.org/10.1093/comjnl/10.1.85