TPLP **21** (5): 527–542, 2021. © The Author(s), 2021. Published by Cambridge University Press
doi:10.1017/S1471068421000247

Modal Logic S5 Satisfiability in Answer Set Programming

MARIO ALVIANO
University of Calabria, Italy
(*e-mail:* alviano@mat.unical.it)

SOTIRIS BATSAKIS
Technical University of Crete, Greece and University of Huddersfield, UK
(*e-mail:* s.batsakis@hud.ac.uk)

GEORGE BARYANNIS
School of Computing and Engineering, University of Huddersfield, UK
(*e-mail:* g.bargiannis@hud.ac.uk)

submitted 10 August 2021; revised 10 August 2021; accepted 21 August 2021

Abstract

Modal logic S5 has attracted significant attention and has led to several practical applications, owing to its simplified approach to dealing with nesting modal operators. Efficient implementations for evaluating satisfiability of S5 formulas commonly rely on Skolemisation to convert them into propositional logic formulas, essentially by introducing copies of propositional atoms for each set of interpretations (possible worlds). This approach is simple, but often results into large formulas that are too difficult to process, and therefore more parsimonious constructions are required. In this work, we propose to use Answer Set Programming for implementing such constructions, and in particular for identifying the propositional atoms that are relevant in every world by means of a reachability relation. The proposed encodings are designed to take advantage of other properties such as entailment relations of subformulas rooted by modal operators. An empirical assessment of the proposed encodings shows that the reachability relation is very effective and leads to comparable performance to a state-of-the-art S5 solver based on SAT, while entailment relations are possibly too expensive to reason about and may result in overhead.

KEYWORDS: modal logic, S5, Answer Set Programming, Kripke semantics

1 Introduction

Modal logics have been extensively studied in the context of knowledge representation and reasoning and, more broadly, artificial intelligence, providing the ability to qualify truth according to different modalities, most commonly the alethic modalities of necessity and possibility. Several practical applications of modal logics have been proposed, such as analysing syntax structures and natural language semantics in linguistics (Moss and Tiede 2007), belief and trust in multi-agent systems (Liau 2003) and weak models of distributed computing (Hella *et al*. 2015). Applications are even more extensive if we consider other modalities such as epistemic logic, used to reason about knowledge and belief states of agents (Moses 2008), deontic logic, used in legal representation and reasoning (Batsakis

https://doi.org/10.1017/S1471068421000247 Published online by Cambridge University Press

et al. 2018) and temporal logics, which form the basis of significant formal verification and model checking approaches, beginning with Amir Pnueli's seminal work (Pnueli 1977).

The main approach for defining the semantics of modal logics is Kripke (or relational) semantics (Kripke 1959), with formulas interpreted according to a set of possible worlds, in which formulas can be true or false, and an accessibility relation that determines which worlds are accessible assuming that we are at one of them. Kripke semantics have been used to model several different modal logic systems, including S5, in which the accessibility relation is actually an equivalence relation, since it satisfies the reflexive, symmetric and transitive properties. An additional attractive feature of S5 is that sequences of modal operators applied on a formula are simplified by pruning all but the one closest to the formula.

Significant research effort has been devoted to solving the satisfiability problem for S5 (S5-SAT), which has been proven to be NP-complete (Ladner 1977), exploring various established proof methods such as tableau (Goré 1999), resolution (Nalon and Dixon 2007) and propositional satisfiability (SAT). As argued by Huang *et al*. (2019), SAT-based methods for modal logic satisfiability have increasingly shown potential, both due to improvements in modern SAT solvers and because of their ability to learn from conflicts and balance between guessing and reasoning. Reducing S5-SAT to SAT commonly involves Skolemisation, introducing fresh Boolean variables to denote truth values in different possible worlds, which often results in quite large formulas that are difficult to process. Recently proposed S5-SAT algorithms tackle this issue in different ways: notably S52SAT (Caridroit *et al*. 2017) relies on improved upper bounds on the number of possible worlds required to find a model as well as structural caching, while S5Cheetah (Huang *et al*. 2019) benefits from formula normalisation and a graph-based representation of conflicts between modalised literals.

In this work, we propose to use Answer Set Programming (ASP) (Lifschitz 2019; Brewka *et al*. 2011) to provide parsimonious propositional representations of S5 formulas and rely on ASP solvers for the satisfiability problem of S5. The motivation behind this proposal lies in the close relationship between ASP and SAT and the readability and configurability afforded by ASP encodings due to their logic programming nature (Baryannis *et al*. 2018; 2020). We first provide a complete ASP encoding of the S5 normal form introduced by Huang *et al*. (2019). Then, we introduce an optimisation to the base ASP encoding exploiting the fact that only some propositional atoms are reachable through propagation based on the existence of the possibility modal operator. We also look at additional modifications based on known entailment and conflict relations between modal operators.

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

- We propose an approach to convert an S5 formula to an ASP program that is applicable to any formula that has been normalised according to the S5 normal form introduced by Huang *et al*. (2019).
- An optimised version of the base ASP encoding is also proposed, that benefits from materialising only those atoms that are reachable. We also explore additional modifications: modal subsets (i.e. detecting if modal operators are applied on literals which are subsets of literals with the same modal operators) and modal conflicts (i.e. detecting if specific combinations of modal operators lead to conflicts).

https://doi.org/10.1017/S1471068421000247 Published online by Cambridge University Press

- S5PY, an ASP-based solver for modal logic S5 satisfiability is implemented and evaluated. The solver automatically converts arbitrary S5 formulas into their equivalent S5 normal forms and then to ASP programs. The solver may apply some of the optimisations/modifications depending on runtime options in order to find a model.

The proposed ASP-based S5-SAT solver is experimentally evaluated to determine the relative efficiency of different encodings and the benefits of each optimisation, confirming that the reachable atoms grounding optimisation achieves the most significant improvement, while modifications based on entailment and conflicts may result in overhead due to them being potentially expensive to reason about. Additionally, we compare different versions of the encoding (including some or all of the modifications) with the state-of-the-art solver S5CHEETAH looking at the trade-off between the number of solved instances and the required execution time showing that the proposed solver achieves comparable performance.

The rest of this paper is organised as follows. Section 2 summarises background knowledge on modal logics, S5 and S5 normal form, as well as related research on S5 satisfiability solving. Section 3 provides a formalisation of the reachability relation for the proposed optimisation and other concepts relevant to the modal subsets and modal conflicts modifications, while Section 4 details all ASP encodings, including the base encoding of an S5 formula in S5 normal form and the proposed optimisations and modifications. Section 5 presents and discusses the implementation and experimental evaluation of the ASP-based S5-SAT solver, Finally, Section 6 concludes and points out future research directions.

2 Background and related work

In this section we briefly summarise preliminaries about modal logic S5, beginning with its syntax and semantics in Section 2.1 and the S5 normal form introduced by Huang *et al.* (2019) in Section 2.2. This is followed by a presentation of existing reasoning approaches for S5 in Section 2.3.

2.1 Modal logic S5

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 $\mathcal{A}$. 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 $p \in \mathcal{A}$, 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 (Fitting 1999), so for the purposes of this paper only the set of worlds will be used. A *world* is an interpretation of propositional logic, that is, a

https://doi.org/10.1017/S1471068421000247 Published online by Cambridge University Press

function I assigning a truth value of either 0 (false) or 1 (true) to every propositional atom in $\mathcal{A}$. 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 $(\mathbf{I}, 0) \models \phi$?

2.2 S5 normal form

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 normal form (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.

Let ϕ, ψ be S5 formulas, and p be a propositional atom. Let $\psi[\phi/p]$ denote the formula obtained by *substituting* every occurrence of p in ψ with ϕ. Every S5 formula ψ can be transformed into an equi-satisfiable S5-NF formula by applying the transformation tr below to its subformulas (Huang *et al.* 2019). Let p be a *fresh* propositional atom (i.e. an atom not occurring in ψ), $n \geq m \geq 1$, and for all $i \in [1..n]$ let ϕ_i be an S5 formula and $\odot_i \in \{\Box, \Diamond\}$. Transformation tr is defined by the following transformation rules:

1. $\neg\neg\phi_1 \overset{tr}{\mapsto} \phi_1$, $\neg(\phi_1 \wedge \cdots \wedge \phi_n) \overset{tr}{\mapsto} \neg\phi_1 \vee \cdots \vee \neg\phi_n$, $\neg(\phi_1 \vee \cdots \vee \phi_n) \overset{tr}{\mapsto} \neg\phi_1 \wedge \cdots \wedge \neg\phi_n$, $\neg\Box\phi_1 \overset{tr}{\mapsto} \Diamond\neg\phi_1$, and $\neg\Diamond\phi_1 \overset{tr}{\mapsto} \Box\neg\phi_1$;
2. $\odot_1 \cdots \odot_n \phi_n \overset{tr}{\mapsto} \odot_n\phi_n$, $\Box(\phi_1 \wedge \cdots \wedge \phi_n) \overset{tr}{\mapsto} \Box\phi_1 \wedge \cdots \wedge \Box\phi_n$, and $\Diamond(\phi_1 \vee \cdots \vee \phi_n) \overset{tr}{\mapsto} \Diamond\phi_1 \vee \cdots \vee \Diamond\phi_n$;
3. $\Box(\phi_1 \vee \cdots \vee \phi_m \vee \odot_{m+1}\phi_{m+1} \vee \cdots \vee \odot_n\phi_n) \overset{tr}{\mapsto} \Box(\phi_1 \vee \cdots \vee \phi_m) \vee \odot_{m+1}\phi_{m+1} \vee \ldots \vee \odot_n\phi_n$;
4. $\Diamond(\phi_1 \wedge \cdots \wedge \phi_m \wedge \odot_{m+1}\phi_{m+1} \wedge \cdots \wedge \odot_n\phi_n) \overset{tr}{\mapsto} \Diamond(\phi_1 \wedge \cdots \wedge \phi_m) \wedge \odot_{m+1}\phi_{m+1} \wedge \ldots \wedge \odot_n\phi_n$;
5. $\psi[\phi_1 \vee \cdots \vee \phi_m \vee (\phi_{m+1} \wedge \cdots \wedge \phi_n)/p'] \overset{tr}{\mapsto} \psi[(\phi_1 \vee \cdots \vee \phi_m \vee p)/p'] \wedge \Box(\neg p \vee \phi_{m+1}) \wedge \cdots \wedge \Box(\neg p \vee \phi_n)$;
6. $\psi[\phi_1 \wedge \cdots \wedge \phi_m \wedge (\phi_{m+1} \vee \cdots \vee \phi_n)/p'] \overset{tr}{\mapsto} \psi[\phi_1 \wedge \cdots \wedge \phi_m \wedge p/p'] \wedge \Box(\neg p \vee \phi_{m+1} \cdots \vee \phi_n)$.

Note that transformation rules in items 1–4 can be applied working top-down on the tree structure of the formula, and since they are local to subformulas we opted for a simpler notation (e.g. the rule for double negation elimination in item 1 is actually $\psi[\neg\neg\phi_1/p'] \overset{tr}{\mapsto} \psi[\phi_1/p']$, that is, all occurrences of $\neg\neg\phi_1$ in the input formula $\psi[\neg\neg\phi_1/p']$ are transformed into ϕ_1). Moreover, the above transformation rules are intended to be applied in the provided order, and in particular rules 5–6 are defined based on this assumption and are applied working bottom-up on the tree structure of the formula

https://doi.org/10.1017/S1471068421000247 Published online by Cambridge University Press

unless the formula is already in S5-NF. In fact, these transformation rules are introducing a fresh propositional atom p to compactly represent a conjunction or a disjunction, like in Tseitin's transformation, but using less clauses as in the algorithm of Plaisted and Greenbaum (1986). This is achieved by exploiting the fact that the formula is already in negation normal form thanks to rules in item 1 (there is no negation in the path connecting the subformula to the root). Also note that after applying rules 5–6, rule 3 becomes applicable again (and possibly in turn the first rule in item 2). In the following, if not otherwise said, formulas are assumed to be in S5-NF.

2.3 Modal logic solvers

As explained in the introduction, modal logics have many practical applications which have led researchers to explore efficient tools for reasoning problems such as satisfiability. Early approaches that are adopted in practice for reasoning over several modal logics are translation to first-order logic (Ohlbach 1991) and resolution-based methods (Auffray and Hebrard 1990) applicable to S4, K, Q, T and K4 modal logics. The KSP solver (Nalon *et al*. 2017) is a recent contribution that adopts the resolution-based method and is applicable to the propositional multimodal logic K_n. Tableau-based reasoners applicable for modal logics include: SPARTACUS (Götzmann *et al*. 2010) for basic modal logic K; LOTREC (Gasquet *et al*. 2005) which covers a wide range of modal and description logics; and the LCK implementation presented by Abate *et al*. (2007) which is also applicable to several modal and temporal logics.

Performance improvements in modern SAT solvers has led to the development of SAT-based solvers for modal logics. Examples include KSAT (Giunchiglia *et al*. 2000) for modal logic K, which is shown to outperform contemporary tableau and translation-based approaches, and KM2SAT (Sebastiani and Vescovi 2009), which is applicable to K_m modal logic and ALC description logic. INKRESAT (Kaminski and Tebbi 2013) is based on an incremental SAT solver and can deal with modal logics K, T, S4 and K4. S52SAT (Caridroit *et al*. 2017) goes one step beyond these solvers as it addresses the satisfiability problem for S5, which is not supported by most of the other SAT-based solvers. In addition, S52SAT estimates an upper bound of possible worlds using the *diamond degree* of a formula and this tight bound offers a noticeable performance gain.

Following this line of research, the SAT-based S5CHEETAH solver (Huang *et al*. 2019) estimates an upper bound on possible worlds by applying the *graph colourability heuristic*, which is used for identifying non interacting $\Diamond$-literals that can be materialised in the same world. S5CHEETAH can be considered a state-of-the-art S5 solver as it outperforms both the LCK implementation by Abate *et al*. (2007) and the S52SAT solver (Caridroit *et al*. 2017). For this reason, our proposed ASP-based solving approach, formalised in the following section and presented in Section 4, is compared against S5CHEETAH in the experiments discussed in Section 5.

3 S5 satisfiability checking via Skolemisation

In this section, we formalise the proposed approach to checking satisfiability of S5-NF formulas on which the ASP encoding in Section 4 is based. S5 formulas, in general, can be transformed into monadic first-order formulas associating every propositional atom p with a (unary) predicate p; arguments of those predicates are worlds, and therefore

https://doi.org/10.1017/S1471068421000247 Published online by Cambridge University Press

variables are universally quantified for $\Box$-literals and existentially quantified for $\Diamond$-literals. Moreover, since we are interested in the satisfiability problem, the existential closure of the formula is actually checked for satisfiability. Finally, existential variables can be eliminated by Skolemisation, that is, by replacing them with fresh constants.

Formally, 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 *Skolemisation* of ϕ, denoted $sk(\phi)$, is defined by the following transformation rules (applied in the given order):

1. $\Box\psi_i^{\Box} \overset{sk}{\mapsto} \forall x\ \psi_i^{\Box}(x)$, for all $i \in [1..m]$;
2. $\Diamond\psi_i^{\Diamond} \overset{sk}{\mapsto} \psi_i^{\Diamond}(i)$, for all $i \in [1..n]$;
3. $p \overset{sk}{\mapsto} p(0)$, for the remaining propositional literals.

Note that the second transformation rule uses the Skolem constant i to Skolemise formula $\exists x\ \psi_i^{\Diamond}(x)$, and the third transformation rule uses the Skolem constant 0 to Skolemise free variables (which are subject to the existential closure). Hence, in order to check satisfiability of an S5-NF formula ϕ, one can equivalently check satisfiability of the *Herbrand expansion* of $sk(\phi)$, in order to take advantage of modern SAT solvers, for instance.

There are several observations on the structure of S5-NF formulas that can be taken into account in order to improve on the set of clauses that result from the Herbrand expansion, and the rest of this section focuses on these. First of all, fresh propositional atoms must be introduced to represent $\Box$- and $\Diamond$-literals, as well as clauses to impose equivalence between every such fresh propositional atom and the associated S5-literal; as already observed in Section 2.2, implications are actually sufficient (Plaisted and Greenbaum 1986). If $b_1, \ldots, b_m$ and $d_1, \ldots, d_n$ are such fresh propositional atoms, the following implications are encoded in clauses:

- $b_i \to \psi_i^{\Box}(0) \wedge \cdots \wedge \psi_i^{\Box}(n) \equiv (b_i \to \psi_i^{\Box}(0)) \wedge \cdots \wedge (b_i \to \psi_i^{\Box}(n))$, for all $i \in [1..m]$;
- $d_i \to \psi_i^{\Diamond}(i)$, for all $i \in [1..n]$.

Let $he(\phi)$ be such a set of clauses, that is, the Herbrand expansion of $sk(\phi)$.

Example 1. Let ϕ be $p \wedge \Box(p \vee q) \wedge (\Diamond(p \wedge q) \vee \Diamond(\neg p \wedge \neg q))$. Hence, $he(\phi)$ is the set of clauses encoding the following formulas: $p(0) \wedge b_1 \wedge (d_1 \vee d_2)$, $b_1 \to p(0) \vee q(0)$, $b_1 \to p(1) \vee q(1)$, $b_1 \to p(2) \vee q(2)$, $d_1 \to p(1) \wedge q(1)$, and $d_2 \to \neg p(2) \wedge \neg q(2)$. There are two lists of distinct worlds that satisfy ϕ, namely $\mathbf{I_1} = [\{p\}, \{p, q\}]$ and $\mathbf{I_2} = [\{p, q\}]$; they are represented by the following models of $he(\phi)$: $\{b_1, d_1, p(0), p(1), q(1)\} \cup X$ and $\{b_1, d_1, p(0), q(0), p(1), q(1)\} \cup X$, where $X \in 2^{\{p(2), q(2)\}}$. ■

Proposition 3.1. *For every S5-NF formula ϕ, $he(\phi)$ is equi-satisfiable to ϕ.*

A second observation concerns the fact that worlds are relevant only if the associated $\Diamond$-literal is true, and therefore for all $i \in [1..m]$ and $j \in [1..n]$, the clause encoding $b_i \to \psi_i^{\Box}(j)$ can be replaced by a clause encoding $b_i \wedge d_j \to \psi_i^{\Box}(j)$. Actually, if $\psi_j^{\Diamond}(0)$ is true, there is again no need to consider world j. Hence, clauses encoding $b_i \to \psi_i^{\Box}(j)$ in $he(\phi)$ are replaced by clauses encoding $b_i \wedge d_j \wedge \neg \mathit{implied}_j \to \psi_i^{\Box}(j)$, $\mathit{implied}_j \leftrightarrow \psi_j^{\Diamond}(0)$ and $\mathit{implied}_j \to d_j$ to obtain $\mathit{full}(\phi)$, where $\mathit{implied}_j$ is a fresh propositional atom.

https://doi.org/10.1017/S1471068421000247 Published online by Cambridge University Press

Example 2 (Continuing Example 1)**.** Clauses in $full(\phi)$ encode the following formulas: $p(0) \wedge b_1 \wedge (d_1 \vee d_2)$, $b_1 \rightarrow p(0) \vee q(0)$, $b_1 \wedge d_1 \wedge \neg implied_1 \rightarrow p(1) \vee q(1)$, $implied_1 \leftrightarrow p(0) \wedge q(0)$, $implied_1 \rightarrow d_1$, $b_1 \wedge d_2 \wedge \neg implied_2 \rightarrow p(2) \vee q(2)$, $implied_2 \leftrightarrow p(0) \wedge q(0)$, $implied_2 \rightarrow d_2$, $d_1 \rightarrow p(1) \wedge q(1)$, and $d_2 \rightarrow \neg p(2) \wedge \neg q(2)$. The models of $full(\phi)$ are $\{b_1, d_1, p(0), p(1), q(1)\} \cup X_2$ and $\{b_1, d_1, implied_1, p(0), q(0)\} \cup X_1 \cup X_2$, where $X_i \in 2^{\{p(i), q(i)\}}$ for $i \in [1..2]$. ■

Theorem 3.1. *For every S5-NF formula ϕ, $full(\phi)$ is equi-satisfiable to $he(\phi)$.*

Proof. We show equi-satisfiablity of $\Gamma = \{b_i \rightarrow \psi_i^{\Box}(0), b_i \rightarrow \psi_i^{\Box}(j)\}$ and $\Gamma' = \{b_i \rightarrow \psi_i^{\Box}(0), b_i \wedge d_j \wedge \neg implied_j \rightarrow \psi_i^{\Box}(j), implied_j \leftrightarrow \psi_j^{\Diamond}(0), implied_j \rightarrow d_j\}$, from which the claim follows. If $I \models \Gamma$, then $I' = (I \cap atoms(\Gamma)) \cup \{implied_j, d_j \mid I \models \psi_j^{\Diamond}(0)\}$ is such that $I' \models \Gamma'$. As for the other direction, let $I \models \Gamma'$ and $I \models b_i \wedge (\neg d_j \vee implied_j)$ (otherwise $I' \models \Gamma'$ by construction). From $b_i \rightarrow \psi_i^{\Box}(0)$ we have that $I \models \psi_i^{\Box}(0)$, and we can copy world 0 into world j to construct a model I' for Γ: $I' = (I \cap atoms(\Gamma)) \setminus \{p(j) \in I\} \cup \{p(j) \mid p(0) \in I\}$. □

The next observation is more involved and regards the possibility for a world associated with a $\Diamond$-literal to reuse the assignment provided by world 0 to satisfy some $\Box$-literals. Any interpretation I satisfying $full(\phi)$ is such that, for all $i \in [1..m]$ and $j \in [1..n]$, $I \models b_i \rightarrow \psi_i^{\Box}(0)$ and $I \models b_i \wedge d_j \wedge \neg implied_j \rightarrow \psi_i^{\Box}(j)$. Hence, for all $i \in [1..m]$ such that $I(b_i) = 1$, $I \models \psi_i^{\Box}(j)$ for $j \in [0..n]$ if j is world 0 or a relevant world (ie. $j \geq 1$ and $I \models d_j \wedge \neg implied_j$). It turns out that world 0 witnesses the possibility to jointly satisfy all true $\Box$-literals, and therefore the other worlds can focus on $\Box$-literals that may be reached by performing unit propagation from the associated $\Diamond$-literals (resulting into a reachability relation). Other $\Box$-literals can be satisfied by reusing the assignment provided by world 0. 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. Let $reach(\phi)$ be obtained from $full(\phi)$ by removing clauses encoding $b_i \wedge d_j \wedge \neg implied_j \rightarrow \psi_i^{\Box}(j)$, for all $i \in [1..m]$ and $j \in [1..n]$ such that $i \notin B_j$.

Example 3. Let ϕ be $\Box(p \vee q) \wedge \Diamond p \wedge \Diamond \neg p$. Clauses in $full(\phi)$ encode the following formulas:

$$\begin{array}{cccc} b_1 \wedge d_1 \wedge d_2 & b_1 \rightarrow p(0) \vee q(0) & d_1 \rightarrow p(1) & d_2 \rightarrow \neg p(2) \\ b_1 \wedge d_1 \wedge \neg implied_1 \rightarrow p(1) \vee q(1) & implied_1 \leftrightarrow p(0) & implied_1 \rightarrow d_1 & \\ b_1 \wedge d_2 \wedge \neg implied_2 \rightarrow p(2) \vee q(2) & implied_2 \leftrightarrow \neg p(0) & implied_2 \rightarrow d_2. & \end{array}$$

In order to construct $reach(\phi)$, we have to compute sets B_1 and B_2; indeed, ϕ contains two $\Diamond$-literals, namely $\Diamond p$ and $\Diamond \neg p$. Let us first determine the sets of reached literals from $\{p\}$ and $\{\neg p\}$ according to (2): $UP(\{p\}) = \{p\}$ — note that $\overline{p} \notin lits(p \vee q)$; $UP(\{\neg p\}) = \{\neg p, q\}$ — note that $\overline{\neg p} \in lits(p \vee q)$ and therefore literals in $lits(p \vee q) \setminus \{\overline{\neg p}\} = \{q\}$ are added to $UP(\{\neg p\})$; $UP(\{\neg p, q\}) = \{\neg p, q\}$ — note that $\overline{q} \notin lits(p \vee q)$ and therefore no other literal is added to $UP(\{\neg p, q\})$. Hence, we have that $UP \Uparrow lits(\{p\}) = \{p\}$ and

https://doi.org/10.1017/S1471068421000247 Published online by Cambridge University Press

$UP \Uparrow lits(\{\neg p\}) = \{\neg p, q\}$. Now using (3), $B_1 = \emptyset$ and $B_2 = \{1\}$, that is, $\Diamond p$ does not interact with the $\Box$-literal, while $\Diamond\neg p$ interacts with the $\Box$-literal. Accordingly, $reach(\phi)$ is obtained from $full(\phi)$ by removing clauses encoding $b_1 \wedge d_1 \wedge \neg implied_1 \rightarrow p(1) \vee q(1)$. 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 $reach(\phi)$ such that $I(q(0)) = 1$, then $I \cup \{q(1)\}$ is a model of $full(\phi)$; similarly, if I is a model of $reach(\phi)$ such that $I(q(0)) = 0$, then $I \setminus \{q(1)\}$ is a model of $full(\phi)$. ■

Theorem 3.2. *For every S5-NF formula ϕ, $reach(\phi)$ is equi-satisfiable to $full(\phi)$.*

Proof. $I \models full(\phi)$ implies $I \models reach(\phi)$ because $reach(\phi) \subseteq full(\phi)$. As for the other direction, let $I \models reach(\phi)$ be such that $I \models b_i \wedge d_j \wedge \neg implied_j \wedge \neg\psi_i^{\Box}(j)$ for some $i \in [1..m]$ and $j \in [1..n]$ such that $i \notin B_j$ (otherwise $I \models full(\phi)$). Since $b_i \rightarrow \psi_i^{\Box}(0)$ belongs to $reach(\phi)$, we have that $I \models \psi_i^{\Box}(0)$, and we can copy a portion of world 0 into world j to construct a model I' for $reach(\phi)$ such that $I' \models \psi_i^{\Box}(j)$:

- $L = atoms(\psi_i^{\Box}(j)) \setminus atoms(\psi_j^{\Diamond}(j))$;
- $I' = I \setminus \{p(j) \in L\} \cup \{p(j) \in L \mid p(0) \in I\}$.

By reiterating the process, we end up with a model of $full(\phi)$. □

Other observations pertain to entailment relations between $\Box$- and $\Diamond$-literals:

1. $\psi_i^{\Box}$ and $\psi_j^{\Diamond}$ cannot be jointly satisfied if $\overline{lits(\psi_i^{\Box})} \subseteq lits(\psi_j^{\Diamond})$;
2. $\psi_i^{\Box}$ entails $\psi_j^{\Box}$ if $lits(\psi_i^{\Box}) \subseteq lits(\psi_j^{\Box})$;
3. $\psi_i^{\Diamond}$ entails $\psi_j^{\Diamond}$ if $lits(\psi_i^{\Diamond}) \supseteq lits(\psi_j^{\Diamond})$.

Example 4. The following are small examples of the above entailment relations:

1. Formula $\Box(\neg p \vee \neg q) \wedge \Diamond(p \wedge q \wedge s)$ is unsatisfiable; note that $\overline{\{\neg p, \neg q\}} \subseteq \{p, q, s\}$.
2. Whenever $(\mathbf{I}, 0) \models \Box(p \vee q)$, also $(\mathbf{I}, 0) \models \Box(p \vee q \vee s)$ holds; note that $\{p, q\} \subseteq \{p, q, s\}$.
3. Similarly, $(\mathbf{I}, 0) \models \Diamond(p \wedge q \wedge s)$ implies $(\mathbf{I}, 0) \models \Diamond(p \wedge q)$; note that $\{p, q, s\} \supseteq \{p, q\}$.

The provided encodings can be enriched to represent such entailment relations. ■

For $\Gamma \in \{full(\phi), reach(\phi)\}$, let $conflicts(\Gamma, \phi)$ be the set of clauses obtained from Γ by adding clauses of the form $\neg b_i \vee \neg d_j$ for all $i \in [1..m]$ and $j \in [1..n]$ such that $\overline{lits(\psi_i^{\Box})} \subseteq lits(\psi_j^{\Diamond})$; let $boxes(\Gamma, \phi)$ be the propositional formula obtained from Γ by adding clauses of the form $\neg b_i \vee b_j$ for all $i \in [1..m]$ and $j \in [1..m]$ such that $lits(\psi_i^{\Box}) \subseteq lits(\psi_j^{\Box})$; let $diamonds(\Gamma, \phi)$ be the propositional formula obtained from Γ by replacing the clauses encoding $implied_j \leftrightarrow \psi_j^{\Diamond}(0)$ by clauses encoding $implied_j \leftrightarrow \psi_j^{\Diamond}(0) \vee d_{i_1} \vee \cdots \vee d_{i_k}$ for all $j \in [1..n]$, where $\{i_1, \ldots, i_k\} = \{i \mid lits(\Gamma_i^{\Diamond}) \supseteq lits(\psi_j^{\Diamond})\}$.

Theorem 3.3. *For every S5-NF formula ϕ, and $\Gamma \in \{full(\phi), reach(\phi)\}$, the following sets of clauses are equi-satisfiable: Γ, $conflicts(\Gamma, \phi)$, $boxes(\Gamma, \phi)$, and $diamonds(\Gamma, \phi)$.*

Proof. Immediate from the entailment relations. □

4 Modal logic encoding in Answer Set Programming

This section presents an ASP implementation of the propositional theories introduced in the previous section. Common to all theories is the relational encoding of the S5-NF formula ϕ to be processed. Specifically, ϕ is encoded by the following facts:

https://doi.org/10.1017/S1471068421000247 Published online by Cambridge University Press

- 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.

Let $\Pi_{re}(\phi)$ denote the *relational encoding* of ϕ, that is, the above facts.

Example 5. Let ϕ be $p \wedge \Box(p \vee q) \wedge (\Diamond(p \wedge q) \vee \Diamond(\neg p \wedge \neg q))$. $\Pi_{re}(\phi)$ contains the following facts:

```
atom(p).          clause(c1).          clause(c2).
clause(c3).
atom(q).      pos_clause(c1,p).    pos_clause(c2,b1).
pos_clause(c3,d1).
box(b1).          diamond(d1).         diamond(d2).
pos_clause(c3,d2).
pos_box(b1,p).  pos_diamond(d1,p).  neg_diamond(d2,p).
pos_box(b1,q).  pos_diamond(d1,q).  neg_diamond(d2,q).
```

■

The basic encoding materialises a full copy of the propositional atoms in all worlds and introduces a world for every $\Diamond$-literal in the input S5-NF formula ϕ. Let Π_{full} be the ASP program comprising the following rules:

```
$r_{0\z}:\quad$ world(D,D) :- diamond(D).
$r_{1\z}:\quad$ {true(X)} :- box(X).
$r_{2\z}:\quad$ {true(X)} :- diamond(X).
$r_{3\z}:\quad$ {true(X)} :- atom(X).
$r_{4\z}:\quad$ {true(X,W)} :- world(W,_), atom(X).
$r_{5\z}:\quad$ :- clause(C); not true(X) : pos_clause(C,X);
    true(X) : neg_clause(C,X).
$r_{6\z}:\quad$ :- box(B), true(B); not true(X) :
    pos_box(B,X); true(X) : neg_box(B,X).
$r_{7\z}:\quad$ :- world(W,D); box(B), true(B), diamond(D),
    true(D), not implied(D);
$\phantom{r_{7\z}:}\quad$    not true(X,W) : pos_box(B,X);
    true(X,W) : neg_box(B,X).
$r_{8\z}:\quad$ implied(D) :- diamond(D);      true(X) :
    pos_diamond(D,X);
$\phantom{r_{7\z}:}\quad$                                  not
    true(X) : neg_diamond(D,X).
$r_{9\z}:\quad$ :- diamond(D), implied(D), not true(D).
```

https://doi.org/10.1017/S1471068421000247 Published online by Cambridge University Press

```
$r_{10}:\quad$ :- pos_diamond(D,X); true(D), not implied(D);
    world(W,D), not true(X,W).
$r_{11}:\quad$ :- neg_diamond(D,X); true(D), not implied(D);
    world(W,D),       true(X,W).
$r_{12}:\quad$ need(W) :- world(W,D), true(D), not implied(D).
$r_{13}:\quad$ :- atom(X), world(W,_), not need(W), true(X,W).
```

Rule r_0 asserts that every $\Diamond$-literal is associated with its own world, and is possibly replaced in other encodings to let some $\Diamond$-literals share the same world. Rules r_2 and r_3 define the search space for $\Box$-literals and $\Diamond$-literals, that is, each of them can be assumed either true or false. Rule r_3 defines the search space for world 0, that is, every propositional atom can be either true or false; this assumes that the set of worlds is non-empty, in line with the assumption made for Kripke structures. similarly, rule r_4 defines the search space for other worlds. Rule r_5 imposes that all S5-clauses of ϕ are satisfied, and rule r_6 requires that every $\Box$-literal $\Box\psi$ assumed to be true is such that ψ is true in world 0; similarly, rule r_7 requires that ψ is true in all worlds associated with $\Diamond$-literals assumed to be true (and not implied). In fact, rule r_8 defines a $\Diamond$-literal as implied if it is true in world 0, and rule r_9 additionally enforces its truth to reduce the search space. Rules r_{10} and r_{11} enforce truth of every $\Diamond$-literal (assumed to be true and not implied) in the associated world, and rule r_{12} defines such worlds as needed; indeed, rule r_{13} enforces falsity of all propositional atoms in worlds that are not needed, again to reduce the search space.

$\Pi_{full} \cup \Pi_{re}(\phi)$ is an ASP implementation of $full(\phi)$ and has some stable model if and only if there is a set $\mathbf{I} = \{I_0, \ldots, I_n\}$ such that $(\mathbf{I}, 0) \models \phi$. However, it materialises several propositional atoms that can be avoided. So, a second encoding can be so designed as to limit propositional atoms in every world by those reachable from the associated diamonds. Let Π_{reach} be the ASP program obtained from Π_{full} by removing rules r_4 and r_7, and by adding the following rules:

```
$r_{14}:\quad$ {true(Y,W)} :- world(W,D), pos_diamond(D,X),
    lrl(X,p,Y,_).
$r_{15}:\quad$ {true(Y,W)} :- world(W,D), neg_diamond(D,X),
    lrl(X,n,Y,_).
$r_{16}:\quad$ lrl(X,pos,X,pos) :- atom(X), pos_diamond(_,X).
$r_{17}:\quad$ lrl(X,neg,X,neg) :- atom(X), neg_diamond(_,X).
$r_{18}:\quad$ lrl(X,PX,Z,pos) :- lrl(X,PX,Y,neg);
    pos_box(B,Y); pos_box(B,Z), Z!=Y.
$r_{19}:\quad$ lrl(X,PX,Z,neg) :- lrl(X,PX,Y,neg);
    pos_box(B,Y); neg_box(B,Z).
$r_{20}:\quad$ lrl(X,PX,Z,pos) :- lrl(X,PX,Y,pos);
    neg_box(B,Y); pos_box(B,Z).
$r_{21}:\quad$ lrl(X,PX,Z,neg) :- lrl(X,PX,Y,pos);
    neg_box(B,Y); neg_box(B,Z), Z!=Y.
$r_{22}:\quad$ lrb(X,P,B) :- lrl(X,P,Y,neg); pos_box(B,Y).
$r_{23}:\quad$ lrb(X,P,B) :- lrl(X,P,Y,pos); neg_box(B,Y).
$r_{24}:\quad$ reach_box(W,B) :- world(W,D), pos_diamond(D,X);
    lrb(X,pos,B).
```

https://doi.org/10.1017/S1471068421000247 Published online by Cambridge University Press

Algorithm 1: ComputeWorlds(ψ: S5-NF formula)

```
1 D := list of ◇-literals occurring in φ, sorted by decreasing size;
2 W := ∅;
3 foreach ψ in D do
4     if there is w ∈ W such that lits(ψ) ⊆ lits(ψ') for all ψ' ∈ w then
5         w := w ∪ {ψ};                                  // add ψ to world w
6     else
7         W := W ∪ {{ψ}};                                // add a new world for ψ
8 return W;
```

```
$r_{25}:\quad$ reach_box(W,B) :- world(W,D), neg_diamond(D,X);
    lrb(X,neg,B).
$r_{26}:\quad$ :- world(W,D), diamond(D), true(D), not
    implied(D); reach_box(W,B);
$\phantom{r_{26}:}\quad$ true(B); not true(X,W) :
    pos_box(B,X); true(X,W) : neg_box(B,X).
```

Above, lrl stands for *literal reaches literal*, and lrb stands for *literal reaches box*. Rules r_{16}–r_{21} define the *reach* relation introduced in Section 3 for literals occurring in some ◇-literal of ϕ — essentially, set $UP(L)$ in (2). Rules r_{22}–r_{23} detect for each propositional literal ℓ the □-literals that contain a literal reached by ℓ, and rules r_{24}–r_{25} computes for every world the □-literals reached by the associated ◇-literals — essentially, sets B_j in (3). Within such relations, the search space of every world is restricted to the reached propositional literals (rules r_{14}–r_{15}), and □-literals are enforced only if actually reached (rule r_{26}). $\Pi_{reach} \cup \Pi_{re}(\phi)$ is an ASP implementation of $reach(\phi)$.

Let $\Pi_{conflicts}(\phi)$ extend Π_{reach} with the rule

```
$r_{27}:\quad$ :- box_diamond_conflict(B,D); true(B), true(D).
```

and the following facts: box_diamond_conflict(b,d') for every □-literal ψ and ◇-literal ψ' occurring in ϕ and such that $\overline{lits(\psi)} \subseteq lits(\psi')$, where b and d' are the identifiers of ψ and ψ'. $\Pi_{conflicts}(\phi) \cup \Pi_{re}(\phi)$ is an ASP implementation of $conflicts(reach(\phi), \phi)$.

Let $\Pi_{boxes}(\phi)$ extend Π_{reach} with the rule

```
$r_{28}:\quad$ :- box_subset(B,B'), true(B), not true(B').
```

and the following facts: box_subset(b,b') for all □-literals ψ, ψ' occurring in ϕ and such that $lits(\psi) \subseteq lits(\psi')$, where b and b' are the identifiers of ψ and ψ'. $\Pi_{boxes}(\phi) \cup \Pi_{re}(\phi)$ is an ASP implementation of $boxes(reach(\phi), \phi)$.

Let $\Pi_{diamonds}(\phi)$ extend $\Pi_{reach} \setminus \{r_0\}$ with the rule

```
$r_{29}:\quad$ implied(D) :- diamond_subset(D,D'), true(D').
```

and the following facts: diamond_subset(d,d') for all ◇-literals ψ, ψ' occurring in ϕ and such that $lits(\psi) \subseteq lits(\psi')$, where d and d' are the identifiers of ψ and ψ'; world(d,d') for all $w \in$ ComputeWorlds(ψ) with largest ◇-literal ψ, and all $\psi' \in w$, where d and d' are the identifiers of ψ and ψ'. $\Pi_{diamonds}(\phi) \cup \Pi_{re}(\phi)$ is an ASP implementation of $diamonds(reach(\phi), \phi)$ with an additional merging of some worlds guided by the entailment relation between ◇-literals.

https://doi.org/10.1017/S1471068421000247 Published online by Cambridge University Press

Table 1. *Overall number of unsolved instances due to timeouts or memory-outs, average execution time (in seconds) and memory consumption (in MB) on solved instances*

Solver (options)	Unsolved	Timeouts	Memory outs	Avg. Time	Avg. Memory
S5PY (FULL)	283	189	94	14.0	512
S5PY (REACH)	15	15	0	12.1	99
S5PY (REACH+ALL)	69	69	0	15.3	86
S5PY (REACH+CONFLICTS)	66	66	0	14.8	90
S5PY (REACH+BOXES)	16	16	0	12.6	99
S5PY (REACH+DIAMONDS)	15	15	0	12.9	96
S5CHEETAH	30	18	12	13.1	345

5 Evaluation

The ASP encodings presented in Section 4 have been implemented into a new solver, S5PY. The solver is written in Python and uses CLINGO version 5.4.0 (Gebser *et al*. 2016) to search for answer sets. This section reports on an empirical comparison between S5PY and S5CHEETAH (Huang *et al*. 2019) on the benchmark used to assess S5CHEETAH. S5PY and pointers to benchmark files are provided at `http://www.mat.unical.it/~alviano/ICLP2021-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; similar limits are used by Huang *et al*. (2019), with memory limit decreased by 1 GB to avoid swapping. For each instance solved within these limits, we measured the execution time and the memory usage.

We tested six configurations of S5PY:

1. FULL, generation of total worlds, i.e. $\Pi_{full} \cup \Pi_{re}(\phi)$);
2. REACH, restriction of each world to reachable propositional atoms, i.e. $\Pi_{reach} \cup \Pi_{re}(\phi)$);
3. REACH+CONFLICTS, use of conflict relation, i.e. $\Pi_{conflicts}(\phi) \cup \Pi_{re}(\phi)$);
4. REACH+BOXES, use of subset relation for $\Box$-literals, i.e. $\Pi_{boxes}(\phi) \cup \Pi_{re}(\phi)$);
5. REACH+DIAMONDS, use of subset relation for $\Diamond$-literals, i.e. $\Pi_{diamonds}(\phi) \cup \Pi_{re}(\phi)$);
6. REACH+ALL, use of the three relations above.

Aggregated results are reported in Table 1, where it is directly evident that the generation of total worlds is often infeasible in practice. We also observe that S5PY (REACH) and S5CHEETAH have a similar performance in terms of solved instances, which confirms that the restriction of each world to reachable propositional literals is a meaningful alternative to the strategy implemented by S5CHEETAH. Finally, we observe that other relations that S5PY can use in its ASP encodings do not provide any performance improvement, and actually the use of the conflict relation has a sensible negative impact.

A cactus plot is shown in Figure 1, where for each solver the solved instances are sorted by increasing execution time. The performance of a virtual best solver is also shown, for which the execution time is the minimum execution time across all solvers. We can observe that there is no particular solver dominating over the others, though S5PY (REACH) achieves the closest performance to the virtual best solver.

An blew instance by instance comparison of the performance of S5PY (REACH) versus S5PY (FULL) and S5CHEETAH is provided in Figure 2, in terms of execution time and

https://doi.org/10.1017/S1471068421000247 Published online by Cambridge University Press

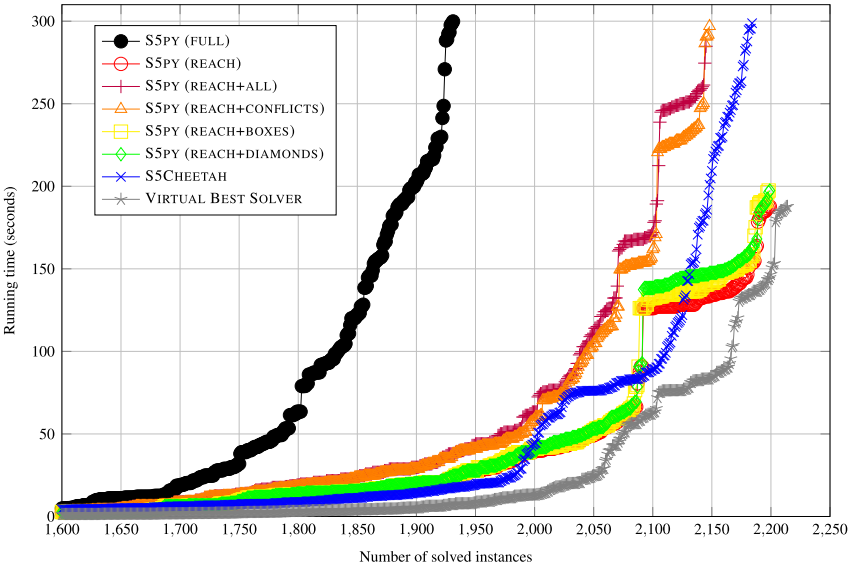


Fig. 1. Number of solved instances within a time budget.

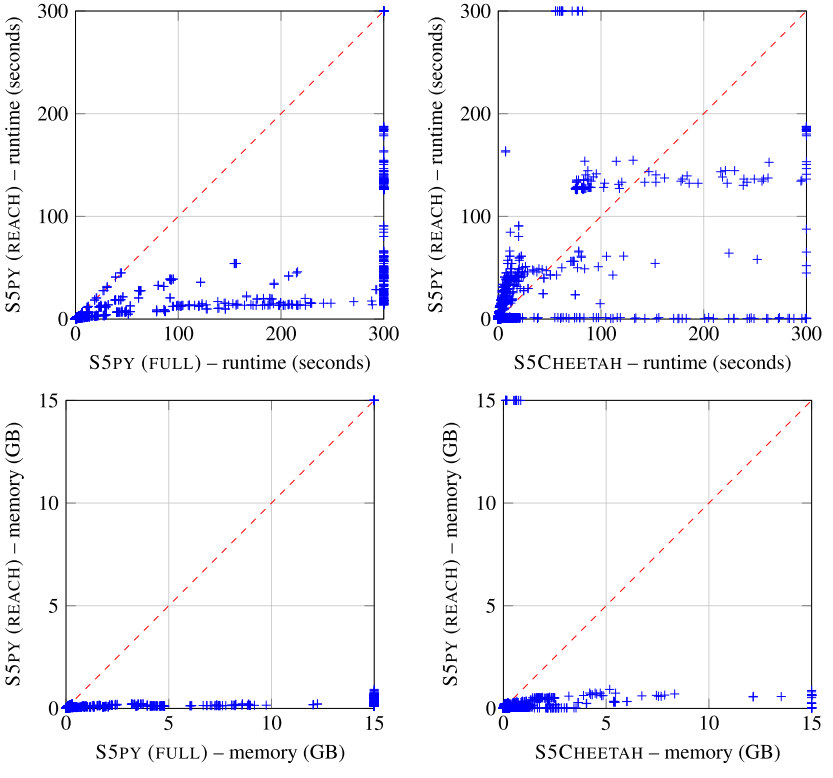


Fig. 2. Instance by instance comparison on execution time and memory: Impact of the reachability relation on S5PY (left) and assessment with respect to the state-of-the-art S5CHEETAH (right) in terms of execution time (top) and memory consumption (bottom). Unsolved instances normalised to the limits.

https://doi.org/10.1017/S1471068421000247 Published online by Cambridge University Press

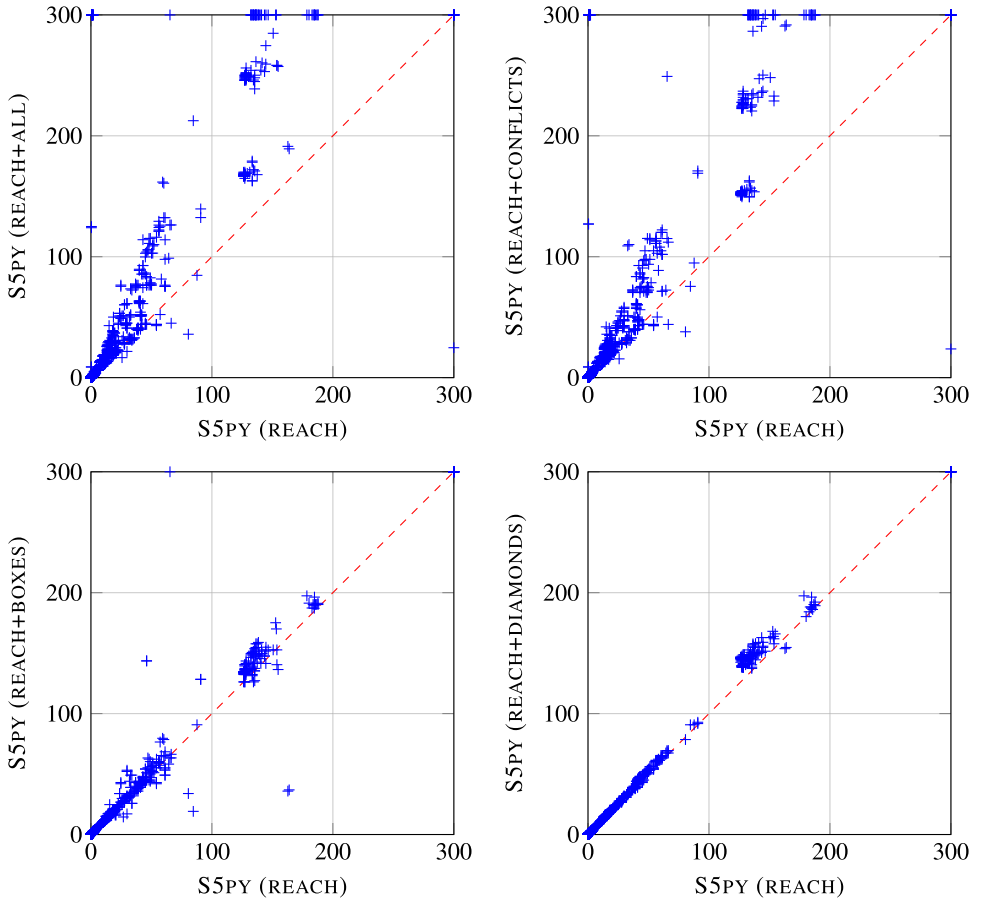


Fig. 3. Instance by instance comparison on execution time (in seconds): Impact of the entailment and conflict relations on S5PY. Unsolved instances normalised to the limit.

memory consumption. It is quite evident that S5PY (REACH) has a uniform and sensible improvement over S5PY (FULL), as all points are essentially below the bisector. Compared to S5CHEETAH, S5PY (REACH) requires less memory in general, as shown in the bottom two plots of Figure 2. However, execution time is not always in its favour, which is also evident from the cactus plot in Figure 1.

Scatter plots shown in Figure 3 confirm that other relations that S5PY can use in its ASP encodings often introduce overhead, since almost all points are above the bisector. The overhead is relatively negligible in the case of S5PY (REACH+BOXES) and S5PY (REACH+DIAMONDS) since the solver has to take into account only one additional subset relation for $\Box$-literals and $\Diamond$-literals, respectively. However, handling the conflict relation imposes a more significant overhead on the solver, since all possible pairs of $\Box$-literals and $\Diamond$-literals need to be considered. Given these results, we conclude that the usage of entailment and conflict relations is unlikely to be justified given its negative impact, while the reachability relation allows S5PY to achieve comparable execution time to S5CHEETAH while consuming less memory.

https://doi.org/10.1017/S1471068421000247 Published online by Cambridge University Press

6 Conclusions and future work

In this work, we have shown that using Answer Set Programming for implementing solvers for modal logic S5 is a both feasible and performant approach. Experimental evaluation of the proposed encodings highlights the performance gain achievable by limiting the $\Box$-literals to satisfy in every world, to those potentially reachable (by unit propagation) from the associated $\Diamond$-literals. In fact, the implemented solver, S5PY, achieves a comparable performance to the state-of-the-art SAT-based solver S5CHEETAH, with none of the two dominating the other.

Future research directions include: (a) considering combinations of the reachability-based optimisation with other optimisations proposed in literature such as the graph colouring approach implemented by S5CHEETAH (Huang *et al.* 2019); (b) defining incremental versions of Π_{reach} to further mitigate the negative impact of oversised propositional formulas; (c) exploring whether optimised implementations can form the basis for solvers in multi-agent settings or for related logics such as modal logic S4 and the implicational fragment of intuitionistic propositional logic (IPC) (Goré and Thomson 2019).

References

ABATE, P., GORÉ, R. AND WIDMANN, F. 2007. Cut-free single-pass tableaux for the logic of common knowledge. In *Workshop on Agents and Deduction at TABLEAUX*.

AUFFRAY, Y. AND HEBRARD, J.-J. 1990. Strategies for modal resolution: results and problems. *Journal of Automated Reasoning 6,* 1, 1–38.

BARYANNIS, G., TACHMAZIDIS, I., BATSAKIS, S., ANTONIOU, G., ALVIANO, M. AND PAPADAKIS, E. 2020. A generalised approach for encoding and reasoning with qualitative theories in answer set programming. *Theory and Practice of Logic Programming 20,* 5, 687–702.

BARYANNIS, G., TACHMAZIDIS, I., BATSAKIS, S., ANTONIOU, G., ALVIANO, M., SELLIS, T. AND TSAI, P.-W. 2018. A trajectory calculus for qualitative spatial reasoning using answer set programming. *Theory and Practice of Logic Programming 18,* 3–4, 355–371.

BATSAKIS, S., BARYANNIS, G., GOVERNATORI, G., TACHMAZIDIS, I. AND ANTONIOU, G. 2018. Legal representation and reasoning in practice: A critical comparison. In *Legal Knowledge and Information Systems - JURIX 2018: The Thirty-first Annual Conference*. IOS Press, Netherlands, 31–40.

BREWKA, G., EITER, T. AND TRUSZCZYŃSKI, M. 2011. Answer set programming at a glance. *Communications of the ACM 54,* 12, 92–103.

CARIDROIT, T., LAGNIEZ, J.-M., BERRE, D. L., DE LIMA, T. AND MONTMIRAIL, V. 2017. A sat-based approach for solving the modal logic s5-satisfiability problem. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, AAAI 2017*. AAAI Press, Palo Alto, California, 3864–3870.

FITTING, M. 1999. A simple propositional S5 tableau system. *Annals of Pure and Applied Logic 96,* 1–3, 107–115.

GASQUET, O., HERZIG, A., LONGIN, D. AND SAHADE, M. 2005. LoTREC: Logical Tableaux Research Engineering Companion. In *Automated Reasoning with Analytic Tableaux and Related Methods*. Springer Berlin Heidelberg, Berlin, Heidelberg, 318–322.

GEBSER, M., KAMINSKI, R., KAUFMANN, B., OSTROWSKI, M., SCHAUB, T. AND WANKO, P. 2016. Theory solving made easy with clingo 5. In *Technical Communications of the Thirty-second International Conference on Logic Programming (ICLP 2016)*, M. Carro and A. King,

https://doi.org/10.1017/S1471068421000247 Published online by Cambridge University Press

Eds. Open Access Series in Informatics (OASIcs), vol. 52. Schloss Dagstuhl, Dagstuhl, Germany, 2:1–2:15.

Giunchiglia, E., Sebastiani, R., Giunchiglia, F. and Tacchella, A. 2000. Sat vs. translation based decision procedures for modal logics: a comparative evaluation. *Journal of Applied Non-Classical Logics 10,* 2, 145–172.

Goré, R. 1999. Tableau methods for modal and temporal logics. In *Handbook of Tableau Methods*, M. D'Agostino, D. M. Gabbay, R. Hähnle and J. Posegga, Eds. Springer, Netherlands, 297–396.

Goré, R. and Thomson, J. 2019. A correct polynomial translation of s4 into intuitionistic logic. *The Journal of Symbolic Logic 84,* 2, 439–451.

Götzmann, D., Kaminski, M. and Smolka, G. 2010. Spartacus: A tableau prover for hybrid logic. *Electronic Notes in Theoretical Computer Science 262*, 127–139. Proceedings of the 6th Workshop on Methods for Modalities (M4M-6 2009).

Hella, L., Järvisalo, M., Kuusisto, A., Laurinharju, J., Lempiäinen, T., Luosto, K., Suomela, J. and Virtema, J. 2015. Weak Models of Distributed Computing, with Connections to Modal Logic. *Distributed Computing 28,* 1, 31–53.

Huang, P., Liu, M., Wang, P., Zhang, W., Ma, F. and Zhang, J. 2019. Solving the satisfiability problem of modal logic S5 guided by graph coloring. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019*, S. Kraus, Ed. ijcai.org, California, 1093–1100.

Kaminski, M. and Tebbi, T. 2013. Inkresat: Modal reasoning via incremental reduction to sat. In *International Conference on Automated Deduction*. Springer, Berlin, Heidelberg, 436–442.

Kripke, S. A. 1959. A completeness theorem in modal logic. *The Journal of Symbolic Logic 24,* 1, 1–14.

Ladner, R. E. 1977. The computational complexity of provability in systems of modal propositional logic. *SIAM Journal on Computing 6,* 3, 467–480.

Liau, C.-J. 2003. Belief, information acquisition, and trust in multi-agent systems a modal logic formulation. *Artificial Intelligence 149,* 1, 31–60.

Lifschitz, V. 2019. *Answer Set Programming*. Springer, Berlin, Heidelberg.

Moses, Y. 2008. Reasoning about knowledge and belief. *Foundations of Artificial Intelligence 3*, 621–647.

Moss, L. S. and Tiede, H.-J. 2007. 19 applications of modal logic in linguistics. In *Handbook of Modal Logic*, P. Blackburn, J. Van Benthem, and F. Wolter, Eds. Studies in Logic and Practical Reasoning, vol. 3. Elsevier, Amsterdam, 1031–1076.

Nalon, C. and Dixon, C. 2007. Clausal resolution for normal modal logics. *Journal of Algorithms 62,* 3, 117–134.

Nalon, C., Hustadt, U., and Dixon, C. 2017. Ksp: A resolution-based prover for multimodal K, abridged report. In *IJCAI*. Vol. 17. ijcai.org, California, 4919–4923.

Ohlbach, H. J. 1991. Semantics-based translation methods for modal logics. *Journal of Logic and Computation 1,* 5, 691–746.

Plaisted, D. A. and Greenbaum, S. 1986. A structure-preserving clause form translation. *Journal of Symbolic Computation 2,* 3, 293–304.

Pnueli, A. 1977. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*. IEEE Computer Society, Los Alamitos, CA, USA, 46–57.

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

https://doi.org/10.1017/S1471068421000247 Published online by Cambridge University Press