Verified Over-Approximation of the Diameter of Propositionally Factored Transition Systems

Mohammad Abdulaziz[†], Charles Gretton[†‡], and Michael Norrish[†]

[†]Canberra Research Lab., NICTA[⋆]; [†]Australian National University; [‡] Griffith University

Abstract. To guarantee the completeness of bounded model checking (BMC) we require a *completeness threshold*. The *diameter* of the Kripke model of the transition system is a valid completeness threshold for BMC of safety properties. The *recurrence diameter* gives us an upper bound on the diameter for use in practice. Transition systems are usually described using (propositionally) factored representations. Bounds for such lifted representations are calculated in a compositional way, by first identifying and bounding atomic subsystems, and then composing those results according to subsystem dependencies to arrive at a bound for the concrete system. Compositional approaches are invalid when using the diameter to bound atomic subsystems, and valid when using the recurrence diameter. We provide a novel overapproximation of the diameter, called the *sublist diameter*, that is tighter than the recurrence diameter. We prove that compositional approaches are valid using it to bound atomic subsystems. Those proofs are mechanised in HOL4. We also describe a novel verified compositional bounding technique which provides tighter overall bounds compared to existing bottom-up approaches.

1 Introduction

Problems in *model checking* and *automated planning* are typically formalised in terms of transition systems. For model checking safety formulae—*i.e.*, globally true formulae of the form $\mathbf{G}p$—one asks: Does every sequence of transitions from the initial state include only states satisfying p? In planning one asks the converse question: Is there a sequence of transitions from a given initial state to a state satisfying a goal condition? In other words, model checking poses a classical planning problem with a goal condition *"p is false"*. For bounded versions of those questions we have an upper bound on the number of transitions that can be taken—*i.e.*, Can the goal be achieved using N transitions? The *diameter* of a system is the length of the longest minimum-length sequence of transitions between any pair of states. Taking "$N =$ diameter" we have that bounded model checking of safety formulae is complete (Biere *et al* [3]). In other words, if the goal condition cannot be reached in N transitions, then it cannot be reached irrespective of the number of transitions taken. In this sense, the diameter is equal to a *completeness threshold* for bounded model checking of safety (Kroening and Strichman [11]).

[⋆] NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program.

A variety of uses for bounds providing the completeness of bounded model checking have been proposed. If the bound N is small, problems reduce in practice to fixed-horizon variants—*i.e.*, is there a sequence of transitions of length less-than-or-equal-to N whose final state satisfies a desired property? That fixed-horizon problem can be posed as a Boolean SAT(isfiability) problem and solved efficiently (Biere *et al* [3], Kautz and Selman [10]). Also, SAT-based approaches which implement query strategies to focus search effort at important horizon lengths, as discussed in Rintanen [14], and Streeter and Smith [19], can also benefit from knowing a small bound. Finally, where system models are given by factored propositional representations and the search goal is to prove that no state satisfying a large conjunct is reachable, by additionally considering the *cones of influence* of variable sets one can use such bounds to identify a small subset of goal propositions which can be easily proved unreachable (see, for example, Rintanen and Gretton [15]).

Despite the important applications above, obtaining useful bounds in practice is not a simple matter. Computing the system diameter is intractable. An encoding of the required formula in quantified Boolean logic to test whether the diameter is N is provided in [3], and is not known to be practically useful. Indeed, for practice the proposal in [3] is to use an overapproximation of the system diameter, namely the *recurrence diameter*. The recurrence diameter corresponds to the length of the longest non-looping path in the Kripke model, thus its computation poses the NP-hard longest path problem. A difficulty in practice is that such structures are described using a compact factored problem description. The Kripke structure corresponding to such a description is much (sometimes exponentially) larger than the description itself, and is rarely explicitly represented. Proposals in this setting employ decompositional bounding procedures which exploit independence between sets of state-characterising variables. Bounds for atomic subsystems are first computed in isolation, which gives the advantage of a potential exponential reduction in the cost of computing the bound on the diameter (*e.g.*, if the recurrence diameter is to be computed). Then the bounds on subsystems are combined either multiplicatively or additively depending on subsystem dependencies. Intuitively, the resulting overapproximation is given by an additive expression whose terms correspond to bounds for independent abstract subproblems. The decompositional approach from Baumgartner *et al* [1] treats design netlists, and in [15] the decomposition treats the *dependency graph* (*causal graph*) associated with a STRIPS-style description. Importantly, decompositional approaches calculate invalid bounds when using the system diameter to bound atomic subproblems. One workaround, as *per* Baumgartner *et al* is to use the recurrence diameter instead.

Treating the *dependency graph* of STRIPS-style descriptions, we have three primary contributions. First, we develop the concept of *sublist diameter* which is a tighter overapproximation of diameter compared to the recurrence diameter. Second, we develop a novel approach to combining the bounds of atomic subproblems, and show that it can provide relatively tight bounds compared to bottom-up approaches, such as those given in [1,15]. Third, we have proved the correctness of both our sublist diameter, and our approach to exploiting it in problem decompositions. Importantly, our proofs are mechanised in the HOL interactive theorem proving system [18].

2 Definitions

Definition 1 (States and Actions). *A transition system is defined in terms of* states *and* actions*: (i) States are finite maps from variables—i.e., state-characterizing propositions—to Booleans, abbreviated as* α state*. We write* $\mathcal{D}(s)$ *for the domain of* s*. (ii) An action* π *is a pair of finite maps over subsets of those variables. The first component of the pair,* $\mathsf{pre}(\pi)$*, is the* precondition *and the second component of the pair,* $\mathsf{eff}(\pi)$*, is the* effect*.*

We give examples of states and actions using sets of literals. For example, $\{x, \neg y, z\}$ is the state where state variables x and z are (map to) true, and y is false.

Definition 2 (Factored Transition System). *A factored transition system* Π *contains two components, (i)* $\Pi.\mathsf{I}$*, a finite map, whose domain is the domain of the system; and (ii)* $\Pi.\mathsf{A}$ *a set of actions, as above. We write* transition-system Π *when the domains of the system's actions' preconditions and effects are all subsets of the system's domain. We also write* $\mathcal{D}(\Pi)$ *for the domain of a system* Π*, and will often take complements of variable sets with respect to it, so that* $\overline{vs}$ *is the set of variables* $\mathcal{D}(\Pi) \setminus vs$*. The set of valid states,* $\mathbb{U}(\Pi)$*, for a factored transition system is* $\{s \mid \mathcal{D}(s) = \mathcal{D}(\Pi)\}$*. The set of valid action sequences,* $\mathbb{A}(\Pi)$*, for a factored transition system is* $\{\dot{\pi} \mid \mathsf{set}\ \dot{\pi} \subseteq \Pi.A\}$*.*

Definition 3 (Action Execution). *When an action* π $(= (p, e))$ *is executed at state* s*, written* $\mathsf{e}(s, \pi)$*, it produces a successor state* s'*. If* p *is not a submap of* s *then* $s' = s$*.*[1] *Otherwise* s' *is a valid state where* $e \sqsubseteq s'$ *and* $s(x) = s'(x)\ \forall x \in \mathcal{D}(s) \setminus \mathcal{D}(e)$*. This operation,* state-succ*, has the following formal definition in HOL:*

```
state-succ s (p,e) = if p ⊑ s then e ⊎ s else s
```

We lift e *to sequences of executions, taking an action sequence* $\dot{\pi}$ *as the second argument. So* $\mathsf{e}(s, \dot{\pi})$ *denotes the state resulting from successively applying each action from* $\dot{\pi}$ *in turn, starting from* s*.*

A Key concept in formalising compositional reasoning about transition systems is *projection*.

Definition 4 (Projection). *Projecting an object (a state* s*, an action* π*, a sequence of actions* $\dot{\pi}$ *or a factored transition system* Π*) on a set of variables* vs *refers to restricting the domain of the object or its constituents to* vs*. We denote these operations as* $s\lfloor_{vs}$*,* $\pi\lfloor_{vs}$*,* $\dot{\pi}\lfloor_{vs}$ *and* $\Pi\lfloor_{vs}$ *for a state, action, action sequence and transition system respectively. HOL provides domain restriction for finite maps in its standard library; projections on composite objects use this operation on the constituents. The one (minor) exception is projection on action sequences, where a projected action is dropped entirely if it has an empty effect:*

```
[]⌊vs = []
((p,e)::π̇)⌊vs = if 𝒟(e⌊vs) ≠ ∅ then (p⌊vs,e⌊vs)::π̇⌊vs else π̇⌊vs
```

[1] Allowing any action to execute in any state is a deviation from standard practice. By using total functions in our formalism, much of the resulting mathematics is relatively simple.

3 Upper Bounding the Diameter with Decomposition

We define the diameter of a transition system in terms of its factored representation as [2]

Definition 5 (Diameter).

$$d(\Pi) = \mathsf{MAX}\ \{\mathsf{MIN}\ (\Pi^d\ (s,\dot{\pi},\Pi)) \mid s \in \mathbb{U}(\Pi) \wedge \dot{\pi} \in \mathbb{A}(\Pi)\}$$

where Π^d *is defined as*

$$\Pi^d\ (s,\dot{\pi},\Pi) = \{|\dot{\pi}'| \mid \mathsf{e}(s,\dot{\pi}') = \mathsf{e}(s,\dot{\pi}) \wedge \dot{\pi}' \in \mathbb{A}(\Pi)\}$$

If the transition system under consideration exhibits a modular or hierarchical structure, upper bounding its diameter compositionally would be of great utility in terms of computational cost; *i.e.*, if the whole system's diameter is bounded by a function of its components' diameters. One source of structure in transition systems, especially hierarchical structure, is *dependency* between state variables.

Definition 6 (Dependency). *A variable* v_2 *is dependent on* v_1 *in a factored transition system* Π *iff one of the following statements holds:*[3] *(i)* v_1 *is the same as* v_2*, (ii) For some action* π *in* A *such that* v_1 *is a precondition of* π *and* v_2 *is an effect of* π*, or (iii) There is an action* π *in* A *such that both* v_1 *and* v_2 *are effects of* π*. We write* $v_1 \rightarrow v_2$ *if* v_1 *is dependent on* v_2 *in* Π*. Formally in HOL we define this as*[4]*:*

$$\begin{array}{l} v_1 \rightarrow v_2 \iff \\ \quad (\exists p\ e\ .\ \\ \qquad (p,e) \in \Pi.\mathsf{A} \wedge \\ \qquad (v_1 \in \mathcal{D}(p) \wedge v_2 \in \mathcal{D}(e) \vee v_1 \in \mathcal{D}(e) \wedge v_2 \in \mathcal{D}(e))) \vee \\ \quad v_1 = v_2 \end{array}$$

We also lift the concept of dependency to sets of variables. A set of variables vs_2 *is dependent on* vs_1 *in a factored transition system* Π *(written* $vs_1 \rightarrow vs_2$*) iff all of the following conditions hold: (i)* vs_1 *is disjoint from* vs_2 *and (ii) There exist variables* $v_1 \in vs_1$ *and* $v_2 \in vs_2$ *such that* $v_1 \rightarrow v_2$*. We define this relation in HOL as*[4]*:*

$$\begin{array}{l} vs_1 \rightarrow vs_2 \iff \\ \quad \exists v_1\ v_2\ .\ v_1 \in vs_1 \wedge v_2 \in vs_2 \wedge \mathsf{DISJOINT}\ vs_1\ vs_2 \wedge v_1 \rightarrow v_2 \end{array}$$

One tool used in analysing dependency structures is the *dependency graph*. The dependency graph of a factored transition system Π is a directed graph, written G, describing variable dependencies. We use it to analyse hierarchical structures in transition systems. This graph was conceived under different guises in [20] and [5], and is also commonly referred to as a *causal graph*.

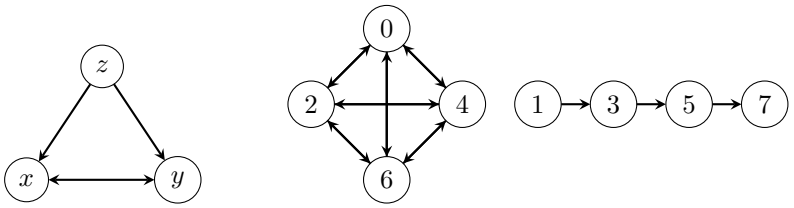
Definition 7 (The Dependency Graph). *The dependency graph has one vertex for each variable in* D*. An edge from* v_1 *to* v_2 *records that* $v_1 \rightarrow v_2$*. When we illustrate a dependency graph we do not draw arcs from a variable to itself although it is dependent on itself.*

---

[2] With this definition the diameter will be one less than how it was defined in [2].

[3] We are using the standard definition of dependency described in [20,5].

[4] $\rightarrow$ has a Π parameter, but we use it with HOL's *ad hoc* overloading ability.



(a) (b)

Fig. 1: a) the dependency and b) the state transition graphs of the system in Example 1

.

We also lift the concept of dependency graphs and refer to lifted *dependency graphs (written G_{vs}). Each node in G_{vs}, for a factored transition system Π, represents a member of a partition of $\mathcal{D}(\Pi)$, i.e. all vertices in G_{vs} represent disjoint sets of variables.*

The structures we aim to exploit are found via dependency graph analysis, where, for example, the lifted dependency graph exhibits a *parent-child* structure, defined as follows:

Definition 8 (Parent-Child Structure). *A system Π has a parent-child structure if $\mathcal{D}(\Pi)$ is comprised of two disjoint sets of variables vs_1 and vs_2 satisfying $vs_2 \not\rightarrow vs_1$. In HOL we model this as follows:*

$$\textsf{child-parent-rel}\ (\Pi, vs) \iff vs \not\rightarrow \overline{vs}$$

It is intuitive to seek an expression that is no worse than multiplication to combine subsystem diameters, or an upper bound for it, into a bound for the entire system's diameter. For instance, for systems with the parent-child structure, previous work in the literature suggests that $(b(\Pi\downarrow_{vs}) + 1)(b(\Pi\downarrow_{\overline{vs}}) + 1)$ should bound $b(\Pi)$, for some upper bound on the diameter $b : \Pi \rightarrow \mathbb{N}$ to be considered *decomposable* for parent-child structures. The following example shows that the diameter is not a decomposable bound for parent-child structures.

Example 1. *Consider a factored transition system Π with the following set of actions*

$$A = \left\{ \begin{array}{l} a = (\{\neg x, \neg y\}, \{x\}), b = (\{x, \neg y\}, \{\neg x, y\}), c = (\{\neg x, y\}, \{x\}), \\ z_1 = (\{\neg z\}, \{x, y\}), z_2 = (\{\neg z\}, \{\neg x, y\}), z_3 = (\{\neg z\}, \{x, \neg y\}), \\ z_4 = (\{\neg z\}, \{\neg x, \neg y\}) \end{array} \right\}.$$

The dependency graph of Π is shown in Figure 1a. $\mathcal{D}(\Pi)$ is comprised of two sets of variables $P = \{z\}$, and the set $C = \{x, y\}$, where $C \not\rightarrow P$ holds. $d(\Pi) = 3$, as this is the length of the longest shortest transition sequence in Π. Specifically, that sequence is $[a; b; c]$ from $\{\neg x, \neg y, z\}$ to $\{x, y, z\}$. $d(\Pi\downarrow_P) = 0$ because in $\Pi\downarrow_P$ the state cannot be changed, and $d(\Pi\downarrow_C) = 1$ because any state in $\Pi\downarrow_C$ is reachable from any other state via one transition with one of the actions $\{z_1\downarrow_C, z_2\downarrow_C, z_3\downarrow_C, z_4\downarrow_C\}$. Accordingly, d is not decomposable.

3.1 Decomposable Diameters

Baumgartner *et al* [1] show that the *recurrence diameter* gives a decomposable upper bound on diameter.

Definition 9 (Recurrence Diameter). *Following Biere* et al *[2], the recurrence diameter is formally defined as follows:*

$$rd(\Pi) = \mathsf{MAX}\ \{|p|\ |\ \mathsf{valid\_path}\ \Pi\ p\ \wedge\ \mathsf{ALL\_DISTINCT}\ p\}$$

where

$$\begin{array}{l}\mathsf{valid\_path}\ \Pi\ [\,] \iff \mathsf{T}\\ \mathsf{valid\_path}\ \Pi\ [s] \iff s \in \mathbb{U}(\Pi)\\ \mathsf{valid\_path}\ \Pi\ (s_1 :: s_2 :: rest) \iff\\ \quad s_1 \in \mathbb{U}(\Pi) \wedge (\exists \pi.\ \pi \in \Pi.\mathsf{A} \wedge \mathsf{e}(s_1,[\pi]) = s_2) \wedge\\ \quad \mathsf{valid\_path}\ \Pi\ (s_2 :: rest)\end{array}$$

We provide a tighter and decomposable diameter: the *sublist diameter*, ℓ.

Definition 10 (Sublist Diameter). *List l_1 is a scattered sublist of l_2 (written $l_1 \preceq l_2$) if all the members of l_1 occur in the same order in l_2. This is defined in HOL as:*

$$\begin{array}{l}[\,] \preceq \ell_1 \iff \mathsf{T}\\ h :: t \preceq [\,] \iff \mathsf{F}\\ x :: \ell_1 \preceq y :: \ell_2 \iff x = y \wedge \ell_1 \preceq \ell_2 \vee x :: \ell_1 \preceq \ell_2\end{array}$$

Based on that the sublist diameter is defined as:

$$\ell(\Pi) = \mathsf{MAX}\ \{\mathsf{MIN}\ \Pi^{\preceq}(s,\dot\pi)\ |\ s \in \mathbb{U}(\Pi) \wedge \dot\pi \in \mathbb{A}(\Pi)\}$$

where $\Pi^{\preceq}$ is defined as

$$\Pi^{\preceq}(s,\dot\pi) = \{|\dot\pi'|\ |\ \mathsf{e}(s,\dot\pi') = \mathsf{e}(s,\dot\pi) \wedge \dot\pi' \preceq \dot\pi\}$$

It should be clear that $\ell(\Pi)$ is an upper bound on $d(\Pi)$ and that it is also a lower bound on $rd(\Pi)$, as demonstrated in the following theorem.

$$\vdash \mathsf{transition\text{-}system}\ \Pi \Rightarrow d(\Pi) \leq \ell(\Pi) \wedge \ell(\Pi) \leq rd(\Pi)$$

The sublist diameter is tighter than the recurrence diameter because it exploits the factored representation of transitions as actions, as shown in the next example.

Example 2. *Consider a factored transition system Π with the following set of actions*

$$A = \left\{ a_1 = (\emptyset, \{x, y\}), a_2 = (\emptyset, \{\neg x, y\}), a_3 = (\emptyset, \{x, \neg y\}), a_4 = (\emptyset, \{\neg x, \neg y\}) \right\}.$$

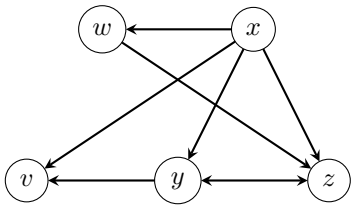
For this system $d(\Pi) = 1$ because any state is reachable from any state with one action. $rd(\Pi) = 3$ because there are many paths with length 3 with no repeated states, but not any longer than that. Lastly, $\ell(\Pi) = 1$ because for any non empty action sequence $\dot\pi \in \mathbb{A}(\Pi)$ the last transition π in $\dot\pi$ can reach the same destination as $\dot\pi$, and $[\pi]$ is a sublist of $\dot\pi$.

$$[\,] \underset{vs}{\wr} \dot{\pi} = \dot{\pi}\downarrow_{D-vs}$$

$$\dot{\pi}_c \underset{vs}{\wr} [\,] = [\,]$$

$$\pi_c :: \dot{\pi}_c \underset{vs}{\wr} \pi :: \dot{\pi} = \begin{cases} \pi :: (\dot{\pi}_c \underset{vs}{\wr} \dot{\pi}) \text{ if } \pi\downarrow_{vs} = \pi_c \\ \pi_c :: \dot{\pi}_c \underset{vs}{\wr} \dot{\pi} \quad \text{o/wise} \end{cases}$$

(a) .



(b)

Fig. 2: (a)The definition of the stitching function ($\wr$), and (b) is the dependency graph of the system in Example 4.

The other important property of ℓ is that it is decomposable as demonstrated in the following example.

Example 3. *Consider the factored transition system Π in Example 1. The values of $\ell(\Pi)$ and $\ell(\Pi\downarrow_P)$ are the same as those of $d(\Pi)$ and $d(\Pi\downarrow_P)$, respectively. However, $\ell(\Pi\downarrow_C) = 3$ because, although any state in $\Pi\downarrow_C$ is reachable from any other state via one transition, there is no shorter sublist of the transition sequence $[a\downarrow_C; b\downarrow_C; c\downarrow_C]$ that starts at $\{\neg x, \neg y\}$ and results in $\{x, y\}$.*

Now we prove that ℓ is decomposable.

Theorem 1. *If the domain of Π is comprised of two disjoint sets of variables vs_1 and vs_2 satisfying $vs_2 \not\rightarrow vs_1$, we have:*

$$\ell(\Pi) < (\ell(\Pi\downarrow_{vs_1}) + 1)(\ell(\Pi\downarrow_{vs_2}) + 1)$$

$$\vdash \textsf{transition-system}\ \Pi\ \wedge\ \textsf{child-parent-rel}\ (\Pi, vs)\ \Rightarrow \\ \ell(\Pi)\ <\ (\ell(\Pi\downarrow_{vs})\ \texttt{+}\ 1)\ \times\ (\ell(\Pi\downarrow_{\overline{vs}})\ \texttt{+}\ 1)$$

Proof. To prove Theorem 1 we use a construction which, given any action sequence $\dot{\pi} \in \mathbb{A}(\Pi)$ violating the stated bound and a state $s \in \mathbb{U}(\Pi)$, produces a sublist, $\dot{\pi}'$, of $\dot{\pi}$ satisfying that bound and $\mathsf{e}(s, \dot{\pi}) = \mathsf{e}(s, \dot{\pi}')$. The premise $vs_2 \not\rightarrow vs_1$ implies that actions with variables from vs_2 in their effects never include vs_1 variables in their effects. Hereupon, if for an action π, $\mathsf{eff}(\pi) \subseteq s$ is true, we call it a s-action. Because vs_1 and vs_2 capture all variables, the effects of vs_2-actions after projection to the set vs_2 are unchanged. Our construction first considers the abstract action sequence $\dot{\pi}\downarrow_{vs_2}$. Definition 10 of ℓ provides a scattered sublist $\dot{\pi}'_{vs_2} \preceq \dot{\pi}\downarrow_{vs_2}$ satisfying $|\dot{\pi}'_{vs_2}| \leq \ell(\Pi\downarrow_{vs_2})$. Moreover, the definition of ℓ can guarantee that $\dot{\pi}'_{vs_2}$ is equivalent, in terms of the execution outcome, to $\dot{\pi}\downarrow_{vs_2}$. The stitching function described in Figure 2a is then used to remove the vs_2-actions in $\dot{\pi}$ whose projections on vs_2 are not in $\dot{\pi}'_{vs_2}$. Thus our construction arrives at the action sequence $\dot{\pi}'' = \dot{\pi}'_{vs_2} \underset{vs_2}{\wr} \dot{\pi}$ with at most $\ell(\Pi\downarrow_{vs_2})$ vs_2-actions. We are left to address the continuous lists of vs_1-actions in $\dot{\pi}''$, to ensure that in

the constructed action sequence any such list satisfies the bound $\ell(\Pi\downarrow_{vs_1})$. The method by which we obtain $\dot{\pi}''$ guarantees that there are at most $\ell(\Pi\downarrow_{vs_2}) + 1$ such lists to address. The definition of ℓ provides that for any abstract list of actions $\dot{\pi}\downarrow_{vs_1}$ in $\Pi\downarrow_{vs_1}$, there is a list that achieves the same outcome of length at most $\ell(\Pi\downarrow_{vs_1})$. Our construction is completed by replacing each continuous sequence of vs_1-actions in $\dot{\pi}''$ with witnesses of appropriate length ($\ell(\Pi\downarrow_{vs_1})$). □

The above construction can be illustrated using the following example.

Example 4. *Consider a factored transition system with the set of actions*

$$A = \left\{ \begin{array}{l} a = (\emptyset, \{x\}), b = (\{x\}, \{y\}), c = (\{x\}, \{\neg v\}), d = (\{x\}, \{w\}), \\ e = (\{y\}, \{v\}), f = (\{w, y\}, \{z\}), g = (\{\neg x\}, \{y, z\}) \end{array} \right\}$$

whose dependency graph is shown in Figure 2b. The domain of Π *is comprised of the two sets* $vs_2 = \{v, y, z\}$ *and* $vs_1 = \{w, x\}$*, where* $vs_2 \not\rightarrow vs_1$*. In* Π*, the actions* b, c, e, f, g *are* vs_2*-actions, and* a, d *are* vs_1*-actions. An action sequence* $\dot{\pi} \in \mathbb{A}(\Pi)$ *is* $[a; a; b; c; d; d; e; f]$ *that reaches the state* $\{v, w, x, y, z\}$ *from* $\{v, \neg w, \neg x, \neg y, \neg z\}$*. When* $\dot{\pi}$ *is projected on* vs_2 *it becomes* $[b\downarrow_{vs_2}; c\downarrow_{vs_2}; e\downarrow_{vs_2}; f\downarrow_{vs_2}]$*, which is in* $\mathbb{A}(\Pi\downarrow_{vs_2})$*. A shorter action sequence,* $\dot{\pi}_c$*, achieving the same result as* $\dot{\pi}\downarrow_{vs_2}$ *is* $[b\downarrow_{vs_2}; f\downarrow_{vs_2}]$*. Since* $\dot{\pi}_c$ *is a scattered sublist of* $\dot{\pi}\downarrow_{vs_2}$*, we can use the stitching function to obtain a shorter action sequence in* $\mathbb{A}(\Pi)$ *that reaches the same state as* $\dot{\pi}$*. In this case,* $\dot{\pi}_c \underset{vs_2}{\between} \dot{\pi}$ *is* $[a; a; b; d; d; f]$*. The second step is to contract the pure* vs_1 *segments which are* $[a; a]$ *and* $[d; d]$*, which are contracted to* $[a]$ *and* $[d]$ *respectively. The final constructed action sequence is* $[a; b; d; f]$*, which achieves the same state as* $\dot{\pi}$*.*

4 Decomposition for Tighter Bounds

So far we have seen how to reason about bounds on underlying transition system sublist diameters by treating sublist diameters of subsystems separately in an example structure (*i.e.*the parent child structure). We now discuss how to exploit a more general dependency structure to compute an upper bound on the sublist diameter of a system, after separately computing subsystems' sublist diameters. We exploit branching one-way variable dependency structures. An example of that type of dependency structure is exhibited in Figure 3a, where S_i are sets of variables each of which forms a node in the lifted dependency graph. Recall that an edge in this graph from a node S_i to a node S_j means $S_i \rightarrow S_j$, which means that there is at least one edge from a variable in S_i to one in S_j. Also, an absence of an edge from a node S_i to a node S_j means that $S_i \not\rightarrow S_j$, and which means that is not a variable in S_j that depends on a variable in S_i.

In this section we present a general theorem about the decompositional properties of the sublist diameter to treat the more general structures. Then we provide a verified upper bounding algorithm based on it. Consider the following more general form of the parent child structure:

Definition 11 (Generalised Parent-Child Structure). *For a factored transition system* Π *and two sets of variables* vs_1 *and* vs_2*, the generalised parent-child relation holds*

between vs_1 and vs_2 iff (i) $vs_2 \not\rightarrow vs_1$, (ii) $vs_1 \not\rightarrow \overline{(vs_1 \cup vs_2)}$, and (iii) no bidirectional dependencies exist between any variable in vs_2 and $\overline{(vs_1 \cup vs_2)}$. Formally, we define this relation in HOL as follows:

$$\begin{array}{l}\mathsf{gen\text{-}parent\text{-}child}\ (\Pi, vs, vs') \iff \\ \quad \mathsf{DISJOINT}\ vs\ vs' \wedge vs' \not\rightarrow vs \wedge vs \not\rightarrow \overline{vs \cup vs'} \wedge \\ \quad \forall v\ v'.\ v \in vs' \wedge v' \in \overline{vs \cup vs'} \Rightarrow v \not\rightarrow v' \vee v' \not\rightarrow v\end{array}$$

To prove the more general theorem we need the following lemma:

Lemma 1. *Let $\mathsf{n}(vs, \dot{\pi})$ be the number of vs-actions contained within $\dot{\pi}$. Consider Π, in which the generalised parent-child relation holds between sets of variables p and c. Then, any action sequence $\dot{\pi}$ has a sublist $\dot{\pi}'$ that reaches the same state as $\dot{\pi}$ starting from any state such that: $\mathsf{n}(p, \dot{\pi}') \leq \ell(\Pi\downarrow_p)(\mathsf{n}(c, \dot{\pi}') + 1)$ and $\mathsf{n}(\overline{p}, \dot{\pi}') \leq \mathsf{n}(\overline{p}, \dot{\pi})$.*

$$\begin{array}{l}\vdash \mathsf{transition\text{-}system}\ \Pi \wedge s \in \mathbb{U}(\Pi) \wedge \dot{\pi} \in \mathbb{A}(\Pi) \wedge \\ \quad \mathsf{gen\text{-}parent\text{-}child}\ (\Pi, p, c) \Rightarrow \\ \quad\quad \exists \dot{\pi}'. \\ \quad\quad\quad \mathsf{n}(p, \dot{\pi}') \leq \ell(\Pi\downarrow_p) \times (\mathsf{n}(c, \dot{\pi}') + 1) \wedge \dot{\pi}' \preceq \dot{\pi} \wedge \\ \quad\quad\quad \mathsf{n}(\overline{p}, \dot{\pi}') \leq \mathsf{n}(\overline{p}, \dot{\pi}) \wedge \mathsf{e}(s, \dot{\pi}) = \mathsf{e}(s, \dot{\pi}')\end{array}$$

Proof. The proof of Lemma 1 is a constructive proof. Let $\dot{\pi}_{\overline{\mathcal{C}}}$ be a contiguous fragment of $\dot{\pi}$ that has no c-actions in it. Then perform the following steps:

- By the definition of ℓ, there must be an action sequence $\dot{\pi}_p$ such that $\mathsf{e}(s, \dot{\pi}_p) = \mathsf{e}(s, \dot{\pi}_{\overline{\mathcal{C}}}\downarrow_p)$, and satisfies $|\dot{\pi}_p| \leq \ell(\Pi\downarrow_p)$ and $\dot{\pi}_p \preceq \dot{\pi}_{\overline{\mathcal{C}}}\downarrow_p$.
- Because $p \not\rightarrow \mathcal{D}(\Pi) \setminus p \setminus c$ holds and using the same argument used in the proof of Theorem 1, $\dot{\pi}'_{\overline{\mathcal{C}}} (= \dot{\pi}_p \overset{\S}{}_p \dot{\pi}_{\overline{\mathcal{C}}}\downarrow_{D\setminus c})$ achieves the same $D \setminus c$ assignment as $\dot{\pi}_{\overline{\mathcal{C}}}$ (*i.e.*, $\mathsf{e}(s, \dot{\pi}'_{\overline{\mathcal{C}}})\downarrow_{D\setminus c} = \mathsf{e}(s, \dot{\pi}_{\overline{\mathcal{C}}})\downarrow_{D\setminus c}$), and it is a sublist of $\dot{\pi}_{\overline{\mathcal{C}}}$. Also, $\mathsf{n}(p, \dot{\pi}'_{\overline{\mathcal{C}}}) \leq \ell(\Pi\downarrow_p)$ holds.
- Finally, because $\dot{\pi}_{\overline{\mathcal{C}}}$ has no c-actions, no c variables change along the execution of $\dot{\pi}_{\overline{\mathcal{C}}}$ and accordingly any c variables in preconditions of actions in $\dot{\pi}_{\overline{\mathcal{C}}}$ always have the same assignment. This means that $\dot{\pi}'_{\overline{\mathcal{C}}} \overset{\S}{}_{D\setminus c} \dot{\pi}_{\overline{\mathcal{C}}}$ will achieve the same result as $\dot{\pi}_{\overline{\mathcal{C}}}$, but with at most $\ell(\Pi\downarrow_p)$ p-actions.

Repeating the previous steps for each $\dot{\pi}_{\overline{\mathcal{C}}}$ fragment in $\dot{\pi}$ yields an action sequence $\dot{\pi}'$ that has at most $\ell(\Pi\downarrow_p)(\mathsf{n}(c, \dot{\pi}) + 1)$ p-actions. Because $\dot{\pi}'$ is the result of consecutive applications of the stitching function, it is a scattered sublist of $\dot{\pi}$. Lastly, because during the previous steps, only p-actions were removed as necessary, the count of the remaining actions in $\dot{\pi}'$ is the same as their number in $\dot{\pi}$. □

Corollary 1. *Let $F(p, c, \dot{\pi})$ be the witness action sequence of Lemma 1. We know then that:*

- $\mathsf{e}(s, F(p, c, \dot{\pi})) = \mathsf{e}(s, \dot{\pi})$,
- $\mathsf{n}(p, F(p, c, \dot{\pi})) \leq \ell(\Pi\downarrow_p)(\mathsf{n}(c, \dot{\pi}) + 1)$.
- $F(p, c, \dot{\pi}) \preceq \dot{\pi}$, *and*
- $\mathsf{n}(\overline{p}, F(p, c, \dot{\pi})) \leq \mathsf{n}(\overline{p}, \dot{\pi})$.

Branching one-way variable dependencies are captured when G_{vs} is a directed acyclic graph (DAG).

Definition 12 (DAG Lifted Dependency Graphs). *In HOL we model a G_{vs} that is a DAG with the predicate* top-sorted *that means that G_{vs} is a list of nodes of a lifted dependency graph topologically sorted w.r.t. dependency. This predicate is defined as follows in HOL:*

$$\begin{array}{l}\mathsf{top\text{-}sorted}\ [\,]\ \Longleftrightarrow\ \mathsf{T}\\ \mathsf{top\text{-}sorted}\ (vs :: G_{vs})\ \Longleftrightarrow\\ \quad (\forall\, vs'.\ vs'\ \in\ \mathsf{set}\ G_{vs}\ \Rightarrow\ vs'\ \not\to\ vs)\ \wedge\ \mathsf{top\text{-}sorted}\ G_{vs}\end{array}$$

We also define the concept of children *for a node vs in G_{vs}, written as $\mathcal{C}(vs)$ to denote the set $\{vs_0 \mid vs_0 \in G_{vs} \wedge vs \to vs_0\}$, which are the children of vs in G_{vs}. In HOL this is modelled as the list:*[5]

$$\mathcal{C}(vs)\ =\ \mathsf{FILTER}\ (\lambda\, vs'.\ vs\ \to\ vs')\ G_{vs}$$

We now use Corollary 1 to prove the following theorem:

Theorem 2. *For a factored transition system Π, and a lifted dependency graph G_{vs} that is a DAG, the sublist diameter $\ell(\Pi)$ satisfies the following inequality:*

$$\ell(\Pi) \leq \Sigma_{vs \in G_{vs}} \mathsf{N}(vs) \tag{1}$$

where $\mathsf{N}(vs) = \ell(\Pi\lfloor_{vs})(\Sigma_{C \in \mathcal{C}(vs)} \mathsf{N}(C) + 1)$.

Alternatively, in the HOL presentation:

$$\begin{array}{l}\vdash\ \mathsf{ALL\_DISTINCT}\ G_{vs}\ \wedge\ \mathsf{ALL\_DISJOINT}\ G_{vs}\ \Rightarrow\\ \quad \forall \Pi\,.\\ \quad\quad \mathsf{transition\text{-}system}\ \Pi\ \wedge\ \mathcal{D}(\Pi\,.\,\mathsf{I})\ =\ \bigcup\ (\mathsf{set}\ G_{vs})\ \wedge\\ \quad\quad \mathsf{top\text{-}sorted}\ G_{vs}\ \Rightarrow\\ \quad\quad\quad \ell(\Pi)\ <\ \mathsf{SUM}\ (\mathsf{MAP}\ \mathsf{N}\ G_{vs})\ +\ 1\end{array}$$

where N is defined as [6]

$$\begin{array}{l}\mathsf{transition\text{-}system}\ \Pi\ \wedge\ \mathsf{top\text{-}sorted}\ G_{vs}\ \Rightarrow\\ \quad \mathsf{N}(vs)\ =\ \ell(\Pi\lfloor_{vs})\ \times\ (\mathsf{SUM}\ (\mathsf{MAP}\ \mathsf{N}\ \mathcal{C}(vs))\ +\ 1)\end{array}$$

Proof. Again, our proof of this theorem follows a constructive approach where we begin by assuming we have an action sequence $\dot{\pi} \in \mathbb{A}(\Pi)$ and a state $s \in \mathbb{U}(\Pi)$. The goal of the proof is to find a witness sublist, $\dot{\pi}'$, of $\dot{\pi}$ such that $\forall\ vs \in G_{vs}.\ \mathsf{n}(vs, \dot{\pi}') \leq \mathsf{N}(vs)$ and $\mathsf{e}(s, \dot{\pi}) = \mathsf{e}(s, \dot{\pi}')$. We proceed by induction on l_{vs}, a topologically sorted list of nodes in G_{vs}. The base case is the empty list $[]$, in which case $\mathcal{D}(\Pi) = \emptyset$ and accordingly $\ell(\Pi) = 0$.

In the step case, we assume the result holds for any system for which l_{vs} is a topologically sorted node list of one of its lifted dependency graphs. We then show that

[5] top-sorted has a Π parameter and $\mathcal{C}$ has Π and G_{vs} as parameters hidden with overloading.

[6] N has Π and G_{vs} as parameters hidden with overloading.

it also holds for Π, a system whose node list is $vs :: l_{vs}$, where vs has no parents (hence its position at the start of the sorted list). Since l_{vs} is a topologically sorted node list of a lifted dependency graph of $\Pi\downarrow_{\overline{vs}}$, the induction hypothesis applies. Accordingly, there is a $\dot{\pi}_{\overline{vs}}$ for $\Pi\downarrow_{\overline{vs}}$ such that $\mathsf{e}(s, \dot{\pi}_{\overline{vs}}) = \mathsf{e}(s, \dot{\pi}\downarrow_{\overline{vs}})$, $\dot{\pi}_{\overline{vs}} \preceq \dot{\pi}\downarrow_{\overline{vs}}$, and $\forall\, K \in l_{vs}.\ \mathsf{n}(K, \dot{\pi}') \leq \mathsf{N}(K)$. Since $vs :: l_{vs}$ is topologically sorted, $(\overline{vs}) \not\rightarrow vs$ holds. Let $\dot{\pi}'_{\overline{vs}} \equiv \dot{\pi}_{\overline{vs}} \,{}^{\S}_{\overline{vs}}\, \dot{\pi}$. Therefore $\mathsf{e}(s, \dot{\pi}'_{\overline{vs}}) = \mathsf{e}(s, \dot{\pi})$ (using the same argument used in the proof of Theorem 1). Furthermore, $\forall K \in l_{vs}.\ \mathsf{n}(K, \dot{\pi}'_{\overline{vs}}) \leq \mathsf{N}(K)$ and $\dot{\pi}'_{\overline{vs}} \preceq \dot{\pi}$. The last step in this proof is to show that $F(vs, \bigcup \mathcal{C}(vs), \dot{\pi}'_{\overline{vs}})$ is the required witness, which is justified because the generalised parent-child relation holds for Π, vs and $\bigcup \mathcal{C}(vs)$. From Corollary 1 and because the relations $=$, $\leq$ and $\preceq$ are transitive, we know that

- $\mathsf{e}(s, \dot{\pi}) = \mathsf{e}(s, F(vs, \bigcup \mathcal{C}(vs), \dot{\pi}'_{\overline{vs}}))$,
- $\mathsf{n}(vs, F(vs, \bigcup \mathcal{C}(vs), \dot{\pi}'_{\overline{vs}})) \leq \ell(\Pi\downarrow_{vs})(\Sigma_{C \in \mathcal{C}(vs)} \mathsf{n}(C, \dot{\pi}'_{\overline{vs}}) + 1)$,
- $F(vs, \bigcup \mathcal{C}(vs), \dot{\pi}'_{\overline{vs}}) \preceq \dot{\pi}$, and
- $\mathsf{n}(\mathcal{D}(\Pi) \setminus vs, F(vs, \bigcup \mathcal{C}(vs), \dot{\pi}'_{\overline{vs}})) \leq \mathsf{n}(\mathcal{D}(\Pi) \setminus vs, \dot{\pi}'_{\overline{vs}})$.

Since $\Sigma_{K \in l_{vs}} \mathsf{n}(K, \dot{\pi}'_{\overline{vs}}) = \mathsf{n}(\mathcal{D}(\Pi) \setminus vs, \dot{\pi}'_{\overline{vs}})$ is true, $\forall\, K \in l_{vs}.\ \mathsf{n}(K, \dot{\pi}'_{\overline{vs}}) \leq \mathsf{N}(K)$ is true, and $\mathsf{n}(vs, F(vs, \bigcup \mathcal{C}(vs), \dot{\pi}'_{\overline{vs}})) \leq \ell(\Pi\downarrow_{vs})(\Sigma_{C \in \mathcal{C}(vs)} \mathsf{N}(C))$ is true, therefore $F(vs, \bigcup \mathcal{C}(vs), \dot{\pi}'_{\overline{vs}})$ is an action sequence demonstrating the needed bound. □

4.1 A Bounding Algorithm

We now discuss an upper bounding algorithm that we prove is valid. Consider the function N_b, defined over the nodes of a lifted dependency DAG as:

$$\mathsf{N}_b(vs) = b(\Pi\downarrow_{vs})(\Sigma_{C \in \mathcal{C}(vs)} \mathsf{N}_b(C) + 1)$$

Note that N_b is a general form of the function N defined in Theorem 2, parameterised over a *base case* function $b : \Pi \to \mathbb{N}$. Viewing $\mathfrak{N}_b = \Sigma_{vs \in G_{vs}} \mathsf{N}_b(vs)$ as an algorithm, the following theorem shows that it calculates a valid upper bound for a factored transition system's sublist diameter if the base case calculation is a valid bound.

Theorem 3. *For a base case function $b : \Pi \to \mathbb{N}$, if $\forall \Pi.\ell(\Pi) \leq b(\Pi)$ then $\ell(\Pi) \leq \mathfrak{N}_b(vs)$.*

$$\begin{array}{l}
\vdash\ (\forall\, \Pi'.\ \textsf{transition-system}\ \ \Pi'\ \Rightarrow\ \ell(\Pi')\ \leq\ b(\Pi'))\ \Rightarrow \\
\quad \forall\, G_{vs}\,. \\
\qquad \textsf{ALL_DISTINCT}\ \ G_{vs}\ \wedge\ \textsf{ALL_DISJOINT}\ \ G_{vs}\ \Rightarrow \\
\qquad\quad \forall\, \Pi\,. \\
\qquad\qquad \textsf{transition-system}\ \ \Pi\ \wedge\ \mathcal{D}(\Pi\,.\,\mathsf{I})\ =\ \bigcup\ (\textsf{set}\ \ G_{vs})\ \wedge \\
\qquad\qquad \textsf{top-sorted}\ \ G_{vs}\ \Rightarrow \\
\qquad\qquad\quad \ell(\Pi)\ <\ \textsf{SUM}\ \ (\textsf{MAP}\ \ \mathsf{N}_b\ \ G_{vs})\ +\ 1
\end{array}$$

where N_b is characterised by the following theorem[7] as

[7] N_b has Π and G_{vs} as parameters hidden with overloading.

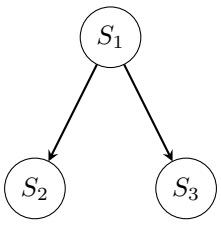
$$
\begin{array}{l}
(\forall\, \Pi' \,.\;\; b(\Pi'\!\downarrow_{\emptyset}) \;=\; 0) \;\Rightarrow \\
\quad \forall\, \Pi \;\; vs \;\; G_{vs} \,. \\
\qquad \mathsf{transition\text{-}system}\;\; \Pi \;\wedge\; \mathsf{top\text{-}sorted}\;\; G_{vs} \;\Rightarrow \\
\qquad\quad \mathsf{N}_b(vs) \;=\; b(\Pi\!\downarrow_{vs}) \;\times\; (\mathsf{SUM}\;\; (\mathsf{MAP}\;\; \mathsf{N}_b\;\; \mathcal{C}(vs)) \;+\; 1)
\end{array}
$$

Without any detailed analysis we are able to take $b(\Pi) = 2^{|\mathcal{D}(\Pi)|} - 1$. In other words, an admissible base case is one less than the number of states representable by the set of state variables of the system being evaluated. That is a valid upper bound for both the recurrence and sublist diameters.

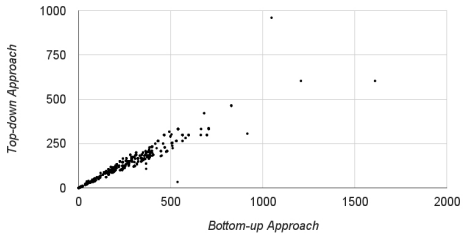
5 Bounds in Practice

In this section we provide an evaluation of the upper bounds produced by the algorithm from Section 4.1. We first compare it to previously suggested compositional bounding approaches in planning benchmarks. We also evaluate its performance on a practical model checking problem.

5.1 Evaluating in Benchmarks from Automated Planning



(a) A lifted dependency graph.



(b) The bounds computed by $\mathfrak{N}_b$ versus $\mathfrak{M}_b$ with $2^{|\mathcal{D}(\Pi)|} - 1$ as a base function.

In this section we compare the bounds computed by $\mathfrak{N}_b$ with the ones computed using the algorithm suggested by Baumgartner *et al* [1] for treating design netlists. This algorithm, and that in Rintanen and Gretton [15], both traverse the structure of the factored transition system in a bottom-up way. By way of contrast, our algorithm traverses the same structure top-down. Before we model the bottom-up calculation, we need to define the concepts of *ancestors* and *leaves*.

Definition 13 (Leaves and Ancestors). *We define the set of leaves $\mathcal{L}(G_{vs})$ to contain those vertices of G_{vs} from which there are no outgoing edges. We also write $\mathcal{A}(vs)$ to denote the set of ancestors of vs in G_{vs} i.e. the set $\{vs_0 \mid vs_0 \in G \wedge vs_0 \rightarrow vs^+\}$, where $\rightarrow^+$ is the transitive closure of $\rightarrow$.*

To model bottom-up calculation, consider the function

$$M_b(vs) = b(\Pi\!\downarrow_{vs}) + (1 + b(\Pi\!\downarrow_{vs}))\Sigma_{A \in \mathcal{A}(vs)} M_b(\Pi\!\downarrow_A)$$

The bottom-up approach, $\mathfrak{M}_b$, can be described as $\mathfrak{M}_b = \Sigma_{vs \in \mathcal{L}(G_{vs})} M_b(vs)$, where b is a base case function.

Consider the lifted dependency DAG in Figure 3a. Given a base a function b, and letting $b(\Pi|_{S_i})$ be b_i, the values of $N_b(S_2)$ and $N_b(S_3)$ are b_2 and b_3, respectively. The value of $N_b(S_1)$ is $b_1 + b_1b_2 + b_1b_3$. Accordingly the value of $\mathfrak{N}_b = b_1 + b_1b_2 + b_1b_3 + b_2 + b_3$.

On the other hand, $M_b(S_1) = b_1$, $M_b(S_2) = b_1 + b_1b_2 + b_2$ and $M_b(S_3) = b_1 + b_1b_3 + b_1$. Accordingly $\mathfrak{M}_b = 2b_1 + b_1b_2 + b_1b_3 + b_2 + b_3$. The value of $\mathfrak{M}_b$ has an extra b_1 term over that of $\mathfrak{N}_b$. This extra term is because $\mathfrak{M}_b$ counts every ancestor node in the lifted dependency graph as many times as the size of its posterity, which is a consequence of the bottom-up traversal of the dependency graph. Figure 3b shows the computed bounds of $\mathfrak{N}_b$ versus $\mathfrak{M}_b$ with the function $2^{|\mathcal{D}(\Pi)|} - 1$ as the base function for a 1030 different International Planning Competition benchmarks. That figure shows that $\mathfrak{M}_b$ computes looser bounds as it repeats counting the ancestor nodes unnecessarily.

5.2 Hotel Key Protocol

$r = g = 5$	190	662	1397	2532	4067	6002	8337	11072	14207	17742
$k = g = 5$	859	1661	2463	3265	4067	4869	5671	6473	7275	8077
$r = k = 5$	803	1619	2435	3251	4067	4883	5699	6515	7331	8147

Table 1: A table showing the bounds computed by $\mathcal{N}_b$ for the hotel key example. Rows 1–3 show the bounds with keeping two of the parameters constant ($= 5$) and the third ranging from 1–10.

We consider the verification of a safety property of the hotel key distribution protocol introduced by Jackson [9] and further discussed in [13,4]. To our knowledge this domain has not previously been explored along with decompositional bounding techniques. We model the problem in the Planning Domain Description Language [12]. There are three actions schemas representing the three categories of change in the system's state which are: (i) entering a room with a new key (`enter-new-key`), (ii) checking in the hotel (`check-in`), and (iii) checking out (`check-out`). Note that we omit the fourth action which is entering the room with its current key, because it has no effect on the system's state. We have a predicate `safe`, that is true of a room while the safety property of that room is maintained. All rooms are initially safe, and then the value of "safe" is set in the effect of `enter-new-key`, and reset in the effect of `check-in`. The goal is to have at least one room that is entered by a guest who does not occupy it and for which "safe" is true.

In our experiment, we parameterised instances of this problem by the number of: guests (g), rooms (r) and keys per room (k). The initial state of an instance asserts:(i) the types of the guests, the rooms and the keys, (ii) which key is owned by which rooms, and (iii) an ordering of the keys such that the keys owned by a room form a series. Table 1 shows the output of N_b on different instances, with an unverified base case

function based on invariants analysis.[8] We computed the sets of variables that satisfy the invariance condition with Fast Downward [8]. Each row shows the computed bounds keeping two of the parameters constant and equal to 5, while the third parameter ranges from 1-10. The computed upper bound increases linearly with the r and with g.

6 Related Work

The notions of diameter and recurrence diameter were introduced in Biere *et al* [3,2]. In this work they describe how to test whether k is the recurrence diameter using a SAT formula of size $O(k^2)$. It was later shown by Kroening and Strichman [11] that this test can be done using a SAT formula of size $O(k \log k)$. An *inductive* algorithm for computing the recurrence diameter was introduced by Sheeran *et al* [17].

Other work exploits the structure or the type of system being verified for efficient computation of the diameter as well as for obtaining tighter bounds on it. For example, Ganai *et al* [7] show that an upper bound on the completeness threshold for checking the safety of some software errors—such as array bound violations—can be computed using a SAT formula of size $O(k)$. Also Konnov *et al* [16] show how some components in threshold-based distributed algorithms have diameters quadratic in the number of transitions in the component. Most relevant to our work, Baumgartner *et al* [1,6] show that the recurrence diameter can be used to calculate a bound for the diameter in a decompositional way using design netlists. In 2013, Rintanen and Gretton [15] described a similar method for calculating a transition system's diameter in the context of planning. The algorithms for calculating a bound in both of those works operate in a bottom-up way.

7 Conclusion

We considered computing admissible completeness thresholds for model checking safety properties in transition models with factored representations. We developed the concept of *sublist diameter*, a novel, tighter overapproximation of diameter for the factored case. We also developed a novel procedure for computing tighter bounds for factored systems by exploiting compositionality, and have formally verified dominance and correctness results associated with these.

The insights which led us to develop the sublist diameter followed from attempting to formalise the results in [15] . That effort helped us find a bug in their formal justification of their approach, where they incorrectly theorise that the diameter can be used directly to bound atomic components for compositional algorithms. Errors such as those make a strong case for the utility of mechanical verification.

In future, we hope to find efficient procedures for efficiently computing/tightly-approximating sublist diameters for the atomic subsystems in our compositional approach.

[8] The point of this experiment is to show how the computed upper bound grows with different parameters, regardless of the base case function used.

HOL4 Notation and Availability All statements appearing with a turnstile ($\vdash$) are HOL4 theorems, automatically pretty-printed to LaTeX. All our HOL scripts, experimental code and data are available from `https://MohammadAbdulaziz@bitbucket.org/MohammadAbdulaziz/planning.git`.

Acknowledgements We thank Daniel Jackson for suggesting applying diameter upper bounding on the hotel key protocol verification.

References

1. Baumgartner, J., Kuehlmann, A., Abraham, J.: Property checking via structural analysis. In: Computer Aided Verification. pp. 151–165. Springer (2002)
2. Biere, A., Cimatti, A., Clarke, E.M., Strichman, O., Zhu, Y.: Bounded model checking. Advances in Computers 58, 117–148 (2003)
3. Biere, A., Cimatti, A., Clarke, E.M., Zhu, Y.: Symbolic model checking without BDDs. In: TACAS. pp. 193–207 (1999)
4. Blanchette, J.C., Nipkow, T.: Nitpick: A counterexample generator for higher-order logic based on a relational model finder. In: Interactive Theorem Proving, First International Conference, ITP 2010. pp. 131–146 (2010)
5. Bylander, T.: The computational complexity of propositional STRIPS planning. Artificial Intelligence 69(1-2), 165–204 (1994)
6. Case, M.L., Mony, H., Baumgartner, J., Kanzelman, R.: Enhanced verification by temporal decomposition. In: Proceedings of 9th International Conference on Formal Methods in Computer-Aided Design, FMCAD 2009, 15-18 November 2009, Austin, Texas, USA. pp. 17–24 (2009)
7. Ganai, M.K., Gupta, A.: Completeness in smt-based BMC for software programs. In: Design, Automation and Test in Europe, DATE 2008, Munich, Germany, March 10-14, 2008. pp. 831–836 (2008)
8. Helmert, M.: The Fast Downward planning system. Journal of Artificial Intelligence Research 26, 191–246 (2006)
9. Jackson, D.: Software Abstractions: Logic, Language, and Analysis. MIT Press (2006)
10. Kautz, H.A., Selman, B.: Planning as satisfiability. In: ECAI. pp. 359–363 (1992)
11. Kroening, D., Strichman, O.: Efficient computation of recurrence diameters. In: VMCAI. pp. 298–309 (2003)
12. Mcdermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., Wilkins, D.: PDDL: The Planning Domain Definition Language. Tech. rep., CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control (1998)
13. Nipkow, T.: Verifying a hotel key card system. In: Barkaoui, K., Cavalcanti, A., Cerone, A. (eds.) Theoretical Aspects of Computing (ICTAC 2006). Lecture Notes in Computer Science, vol. 4281. Springer (2006), invited paper.
14. Rintanen, J.: Evaluation strategies for planning as satisfiability. In: Proc. 16th European Conf. on Artificial Intelligence. pp. 682–687. IOS Press (2004)
15. Rintanen, J., Gretton, C.O.: Computing upper bounds on lengths of transition sequences. In: International Joint Conference on Artificial Intelligence (2013)
16. Sastry, S., Widder, J.: Solvability-based comparison of failure detectors. In: 2014 IEEE 13th International Symposium on Network Computing and Applications, NCA 2014, Cambridge, MA, USA, 21-23 August, 2014. pp. 269–276 (2014)

17. Sheeran, M., Singh, S., Stålmarck, G.: Checking safety properties using induction and a sat-solver. In: Formal Methods in Computer-Aided Design, Third International Conference, FMCAD 2000, Austin, Texas, USA, November 1-3, 2000, Proceedings. pp. 108–125 (2000)
18. Slind, K., Norrish, M.: A brief overview of HOL4. In: Theorem Proving in Higher Order Logics. LNCS, vol. 5170, pp. 28–32. Springer (2008)
19. Streeter, M.J., Smith, S.F.: Using decision procedures efficiently for optimization. In: Proc. 17th International Conference on Automated Planning and Scheduling. pp. 312–319. AAAI Press (2007)
20. Williams, B.C., Nayak, P.P.: A reactive planner for a model-based executive. In: International Joint Conference on Artificial Intelligence. pp. 1178–1185. Morgan Kaufmann Publishers (1997)