Extracting Symbolic Transitions from TLA^+ Specifications * **

Jure Kukovec[1], Thanh-Hai Tran[1], and Igor Konnov[1,2]✉

[1] TU Wien (Vienna University of Technology), Austria
{jkukovec,tran,konnov}@forsyte.at
[2] University of Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France
igor.konnov@inria.fr

Abstract. In TLA^+, a system specification is written as a logical formula that restricts the system behavior. As a logic, TLA^+ does not have assignments and other imperative statements that are used by model checkers to compute the successor states of a system state. Model checkers compute successors either explicitly — by evaluating program statements — or symbolically — by translating program statements to an SMT formula and checking its satisfiability. To efficiently enumerate the successors, TLA's model checker TLC introduces side effects. For instance, an equality $x' = e$ is interpreted as an assignment of e to the yet unbound variable x.
Inspired by TLC, we introduce an automatic technique for discovering expressions in TLA^+ formulas such as $x' = e$ and $x' \in \{e_1, \ldots, e_k\}$ that can be provably used as assignments. In contrast to TLC, our technique does not explicitly evaluate expressions, but it reduces the problem of finding assignments to the satisfiability of an SMT formula. Hence, we give a way to slice a TLA^+ formula in symbolic transitions, which can be used as an input to a symbolic model checker. Our prototype implementation successfully extracts symbolic transitions from a few TLA^+ benchmarks.

1 Introduction

TLA is a general language introduced by Leslie Lamport for specifying temporal behavior of computer systems. It was later extended to TLA^+ [18], which provides the user with a concrete syntax for writing expressions over sets, functions, integers, sequences, etc. TLA^+ does not fix a model of computation, and thus it found applications in the design of concurrent and distributed systems, e.g., see [12,23,24,22,2].

A specification alone brings almost no guarantees of system correctness. As it is an untyped language, TLA^+ allows for expressions such as $1 \cup \{2\}$, which are

* Supported by the Vienna Science and Technology Fund (WWTF) through project APALACHE (ICT15-103) and the Austrian Science Fund (FWF) through Doctoral College LogiCS (W1255-N23).

** This is an extended version of the paper that appears at the 6th International ABZ Conference ASM, Alloy, B, TLA, VDM, Z, 2018. The detailed proofs can be found in Appendix B.

$$
\begin{array}{l}
\textsc{module}\ \textit{prodcons} \\
\textsc{variable}\ S,\ \textit{empty} \\
\textit{Init} \triangleq S = \{\} \land \textit{empty} = \textsc{true} \\
\textit{Produce} \triangleq \land\ \textit{empty}' = \textsc{false} \\
\qquad\qquad\quad \land\ \exists X \in \textsc{subset}\ \{\text{“A”},\ \text{“B”},\ \text{“Z”},\ \text{“1”},\ \text{“8”}\} : S' = S \cup \{X\} \\
\textit{Consume} \triangleq \neg \textit{empty} \land S' \in \textsc{subset}\ S \land \textit{empty}' = (S' = \{\}) \\
\textit{Next} \triangleq \textit{Produce} \lor \textit{Consume}
\end{array}
$$

Fig. 1: A simple producer-consumer

considered ill-typed in statically-typed programming languages. To formally prove specification properties such as safety and liveness, one can use TLAPS — a proof system for TLA$^+$ [8]. Although progress towards proof automation was made in the last years [20], writing formal proofs is still a challenging task [23,24].

On the other side of the spectrum are model checkers that require little user effort to run. Indeed, TLA$^+$ users debug their specifications with TLC [26]. Beyond simple debugging, TLC found serious bugs in specifications of distributed algorithms [23]. Although TLC contains remarkable engineering solutions, its core techniques *enumerate* reachable states and inevitably suffer from state explosion.

Instead of enumerating states, software model checkers run SAT and SMT solvers in the background to reason about computations symbolically. To name a few, CBMC [15] and CPAChecker[3] implement bounded model checking [4] and CEGAR [9]. Domain-specific tools ByMC and Cubicle prove properties of parameterized distributed algorithms with SMT [10,14].

A simple example in Figure 1 illustrates the problems that one faces when developing a symbolic model checker for TLA$^+$. In this example, we model two processes: *Producer* that inserts a subset of $\{\text{“}A\text{”}, \text{“}B\text{”}, \text{“}Z\text{”}, \text{“1”}, \text{“8”}\}$ into the set S, and *Consumer* that removes from S its arbitrary subset. The system is initialized with the operator *Init*. A system transition is specified with the operator *Next* that is defined via a disjunction of operators *Produce* and *Consume*. Both Producer and Consumer maintain the state invariant $\textit{empty} \Leftrightarrow (S = \emptyset)$. We notice the following challenges for a symbolic approach:

1. The specification does not have types. This is not a problem for TLC, since it constructs states on the fly and hence dynamically computes types. In the symbolic case, one can use type synthesis [20] or the untyped SMT encoding [21].
2. Direct translation of *Next* to SMT would produce a *monolithic* formula, e.g., it would not analyze *Produce* and *Consume* as independent actions. This is in sharp contrast to translation of imperative programs, in which variable assignments allow a model checker to focus only on the local state changes.

In this paper, we focus on the second problem. Our motivation comes from the observation on how TLC computes the successors of a given state [18, Ch. 14]. Instead of precomputing all potential successors — which would be anyway impossible without types — and evaluating *Next* on them, TLC explores subformulas of *Next*. The essential exploration rules are: (1) Disjunctions and conjunctions are evaluated from left to right, (2) an equality $x' = e$ assigns the value of e to x' if x' is yet unbound, (3) if an unbound variable appears on the right-hand side of an assignment or in a non-assignment expression, TLC terminates with an error, and (4) operands of a disjunction assign values to the variables independently. In more detail, rule (4) means that whenever a disjunction $A \vee B$ is evaluated and x' is assigned a value in A, this value does not propagate to B; moreover, x' must be assigned a value in B.

In our example, TLC evaluates the actions *Produce* and *Consume* independently and assigns variables as prescribed by these formulas. As TLC is explicit, for each state, it produces at most 2^{2^5} successors in *Produce* as well as in *Consume*.

We introduce a technique to statically label expressions in a TLA$^+$ formula ϕ as assignments to the variables from a set V', while fulfilling the following:

1. For purely Boolean formulas, if one transforms ϕ into an equivalent formula $\bigvee_{1 \leq i \leq k} D_i$ in disjunctive normal form (DNF), then every disjunct D_i has *exactly one* assignment per variable from V'.
2. The assignments adhere the following partial order: if $x' \in V'$ is assigned a value in expression e, that uses a variable $y' \in V'$, then the assignment to y' precedes the assignment to x'.
3. In general, we formalize the above idea with the notion of a branch.

As expected, the following sequence of expressions is given as assignments in our example: (1) $\mathit{empty}' = \text{TRUE}$, (2) $S' = S \cup \{X\}$, (3) $S' \in \text{SUBSET}\, S$, and (4) $\mathit{empty}' = (S' = \emptyset)$. Using this sequence, our technique constructs two symbolic transitions that are equivalent to the actions *Produce* and *Consume*.

In general, finding assignments and slicing a formula into symbolic transitions is not as easy as in our example, because of quantifiers and IF-THEN-ELSE complicating matters. In this paper, we present our solution, demonstrate its soundness and report on preliminary experiments.

2 Abstract Syntax α-TLA$^+$

TLA$^+$ has rich syntax [18], which cannot be defined in this paper. To focus only on the expressions that are essential for finding assignments in a formula, we define abstract syntax for TLA$^+$ formulas below. In our syntax, the essential operators such as conjunctions and disjunctions are included explicitly, while the other non-essential operators are hidden under the star expression $\star$.

We assume predefined three infinite sets:

- A set $\mathcal{L}$ of *labels*. We use notation ℓ_i to refer to its elements, for $i \in \mathbb{N}$.
- A set *Vars*$'$ of *primed variables* that are decorated with prime, e.g., x' and a'.
- A set *Bound* of *bound variables*, which are used by quantifiers.

$$\mathit{Next} \triangleq \ell_1 :: \Big(\ell_2 :: (\ell_3 :: \mathit{empty}' \in \ell_4 :: \star \land \ell_5 :: \exists X \in \ell_6 :: \star : \ell_7 :: S' \in \ell_8 :: \star) \lor \ell_9 :: (\ell_{10} :: \star \land \ell_{11} :: S' \in \ell_{12} :: \star \land \ell_{13} :: \mathit{empty}' \in \ell_{14} :: \star(S'))\Big)$$

Fig. 2: The *Next* operator of producer-consumer in α-TLA$^+$

The abstract syntax α-TLA$^+$ is defined in terms of the following grammar:

$$\begin{aligned}
\mathit{expr} &::= \mathit{ex}_\alpha \mid \ell :: \text{FALSE} \\
&\mid \ell :: v' \in \mathit{ex}_\alpha \mid \ell :: \mathit{expr} \land \cdots \land \mathit{expr} \mid \ell :: \mathit{expr} \lor \cdots \lor \mathit{expr} \\
&\mid \ell :: \exists x \in \mathit{ex}_\alpha :\ \mathit{expr} \mid \ell :: \text{IF } \mathit{ex}_\alpha \text{ THEN } \mathit{expr} \text{ ELSE } \mathit{expr} \\
\mathit{ex}_\alpha &::= \ell :: \star(v', \dots, v') \\
\ell &::= \text{a unique label from the set } \mathcal{L} \\
v' &::= \text{a variable name from the set } \mathit{Vars}' \\
x &::= \text{a variable name from the set } \mathit{Bound}
\end{aligned}$$

A few comments on the syntax and its relation to TLA$^+$ expressions are in order. We require every expression to carry a unique label $\ell_i \in \mathcal{L}$. Although this is not a requirement in TLA$^+$, it is easy to decorate every expression with a unique label. The expressions of the form $\ell :: v' \in \mathit{expr}$ are of ultimate interest to us, as they are treated as assignment candidates. Under certain conditions, they can be used to assign to v' a value from the set represented by the expression *expr*. Perhaps somewhat unexpectedly, expressions such as $v' = e$ and UNCHANGED $\langle v_1, \dots, v_k\rangle$ are not included in our syntax. To keep the syntax minimal, we represent them with $\ell :: v' \in \mathit{expr}$. Indeed, these expressions can be rewritten in an equivalent form: $v' = e$ as $v' \in \{e\}$, and UNCHANGED $\langle v_1, \dots, v_k\rangle$ as $v_1' \in \{v_1\} \land \cdots \land v_k' \in \{v_k\}$. Every non-essential TLA$^+$ expression e is presented in the abstract form $\ell :: \star(v_1', \dots, v_k')$, where $v_1', \dots, v_k'$ are the names of the primed variables that appear in e. When no primed variable appears in an expression, we omit parenthesis and write $\ell :: \star$. TLA$^+$ expressions often refer to user-defined operators, which are not present in our abstract syntax. We simply assume that all non-recursive user-defined operators have been expanded, that is, recursively replaced with their bodies. All uses of recursive operators are hidden under $\star$; hence, recursive operator definitions are ignored when searching for assignment candidates.

It should be now straightforward to see how one could translate a TLA$^+$ expression to our abstract syntax. We write $\alpha(e)$ to denote the expression in α-TLA$^+$, that represents an expression e in the complete TLA$^+$ syntax. With γ we denote the reverse translation from α-TLA$^+$ to TLA$^+$ that has the property that $\gamma(\alpha(e)) = e$. Figure 2 shows the abstract expression $\alpha(\mathit{Next})$ of the operator *Next* defined in Figure 1.

Discussions. Notice that α-TLA$^+$ is missing several fundamental constructs permitted in TLA$^+$, such as CASE expressions, universal quantifiers, and negations. They are all abstracted to $\star$. The primary purpose of α-TLA$^+$ is to allow us to determine whether a given expression containing set inclusion — or equality — can be used as an assignment. If such an expression occurs under a universal quantifier, it is not clear which value should be used for an assignment. Hence, we abstract the expressions under universal quantifiers. For similar reason, we abstract the expressions under negation. The latter is consistent with TLC, which produces an error when given, for example, $Next == \neg(x' = 1)$. Finally, we abstract CASE, due to its semantics, which is defined in terms of the CHOOSE operator [18, Ch. 6]. In practice, there are no potential assignments under CASE in the standard TLA$^+$ examples.

3 Preliminary Definitions

Every TLA$^+$ specification declares a certain finite set of variables, which may appear in the formulas contained therein. Let ϕ be an α-TLA$^+$ expression. We assume, for the purposes of our analysis, that ϕ is associated with some finite set $Vars'(\phi)$, which is a subset of $Vars'$, containing all of the variables that appear in ϕ (and possibly additional ones). This is the set of variables declared by the specification in which $\gamma(\phi)$ appears.

Since the labels are unique, we write $\mathrm{lab}(\ell :: \psi)$ to refer to the expression label ℓ and $\mathrm{expr}(\ell)$ to refer to the expression that is labeled with ℓ. We refer to the set of all subexpressions of ϕ by $\mathrm{Sub}(\phi)$. See [16] for a formal definition.

The set $\mathrm{Sub}(\phi)$ allows us to reason about terms that appear inside an expression ϕ, at some unknown/irrelevant depth. We will often refer to the set of all labels appearing in ϕ, that is, $\mathrm{Labs}(\phi) = \{\mathrm{lab}(\psi) \mid \psi \in \mathrm{Sub}(\phi)\}$.

Of special interest to us are *assignment candidates*, i.e., expressions of the form $\ell :: v' \in \phi_1$. Given a variable $v' \in Vars'(\phi)$ and an α-TLA$^+$ expression ϕ, we write $\mathrm{cand}(v', \phi)$ to mean the set of labels that belong to assignment candidates for v' in subexpressions of ϕ. More formally, $\mathrm{cand}(v', \phi)$ is $\{\ell \mid (\ell :: v' \in \psi) \in \mathrm{Sub}(\phi)\}$. An exhaustive definition is included in [16]. We use the notation $\mathrm{cand}(\phi)$ to mean $\bigcup_{v' \in Vars'(\phi)} \mathrm{cand}(v', \phi)$.

Finally, we assign to each label ℓ in $\mathrm{Labs}(\phi)$ a set $\mathrm{frozen}_\phi(\ell) \subseteq Vars'(\phi)$. Intuitively, if a variable v' is in $\mathrm{frozen}_\phi(\ell)$, then no expression of the form $\hat{\ell} :: v' \in \psi$ can be treated as an assignment inside $\mathrm{expr}(\ell)$. Formally, for every $\ell \in \mathrm{Labs}(\phi)$ the set $\mathrm{frozen}_\phi(\ell)$ is defined as the minimal set satisfying all the constraints in Table 1.

The sets frozen_ϕ naturally lead to the dependency relations $\lhd_{v'}$ on $\mathrm{Labs}(\phi)$, where $v' \in Vars'(\phi)$. We will use $\ell_1 \lhd_{v'} \ell_2$ to mean that ℓ_1 is an assignment candidate for v', which also belongs to the frozen set of ℓ_2. Formally:

$$\ell_1 \lhd_{v'} \ell_2 \iff \ell_1 \in \mathrm{cand}(v', \phi) \land v' \in \mathrm{frozen}_\phi(\ell_2)$$

Intuitively, if $\ell_1 \lhd_{v'} \ell_2$ we want to make sure that $\mathrm{expr}(\ell_1)$ is evaluated before $\mathrm{expr}(\ell_2)$, if possible.

Example 1. Let us look at the following α-TLA$^+$ expression:

$$\ell_1 :: [\exists i \in [\ell_2 :: \star(y')] \colon \ell_3 :: x' \in [\ell_4 :: \star]]$$

Table 1: The constraints on frozen_ϕ

α-**TLA**$^+$ **expression** ϕ	**Constraints on** frozen_ϕ
$\ell :: \star(v'_1, \dots, v'_k)$	$\{v'_1, \dots, v'_k\} \subseteq \text{frozen}_\phi(\ell)$
$\ell :: v' \in \phi_1$	$\text{frozen}_\phi(\ell) = \text{frozen}_\phi(\text{lab}(\phi_1))$
$\ell :: \bigwedge_{i=1}^s \phi_i$ or $\ell :: \bigvee_{i=1}^s \phi_i$	$\text{frozen}_\phi(\ell) \subseteq \text{frozen}_\phi(\text{lab}(\phi_i))$ for $i \in \{1, \dots, s\}$
$\ell :: \exists x \in \phi_1 : \ \phi_2$	$\text{frozen}_\phi(\ell) \subseteq \text{frozen}_\phi(\text{lab}(\phi_1)) \subseteq \text{frozen}_\phi(\text{lab}(\phi_2))$
$\ell :: \text{IF } \phi_1 \text{ THEN } \phi_2 \text{ ELSE } \phi_3$	$\text{frozen}_\phi(\ell) \subseteq \text{frozen}_\phi(\text{lab}(\phi_1))$
	$\text{frozen}_\phi(\text{lab}(\phi_1)) \subseteq \text{frozen}_\phi(\text{lab}(\phi_i))$ for $i = 2, 3$

Take the subexpression $\ell_3 :: x' \in [\ell_4 :: \star]$, which we name ψ . By solving the constraints for $\text{frozen}_\psi(\ell_3)$ we conclude that $\text{frozen}_\psi(\ell_3) = \emptyset$. However, if we take the additional constraints for $\text{frozen}_\phi(\ell_3)$ into consideration, the empty set no longer satisfies all of them, specifically, it does not satisfy the condition imposed by the existential quantifier in ℓ_1. The additional requirement $\{y'\} \subseteq \text{frozen}_\phi(\ell_3)$ implies that $\text{frozen}_\phi(\ell_3) = \{y'\}$. This corresponds to the intuition that expressions under a quantifier, like ψ, implicitly depend on the bound variable and the expressions used to define it, which is $\text{expr}(\ell_2)$ in our example. ◁

4 Formalizing Symbolic Assignments

As TLC evaluates formulas in a left-to-right order, there is a very clear notion of an assignment; the first occurrence of an expression $v' \in S$ is interpreted as an assignment to v'. In our work, we want to *statically* find expressions that can safely be used as assignments. If we were only dealing with Boolean formulas, we could transform the original TLA$^+$ formula to DNF, $\bigvee_{i=1}^s D_i$, and treat each D_i independently. However, we also need to find assignments, which may be nested under existential quantifiers. To transfer our intuition about DNF to the general case we first introduce a transformation boolForm, that captures the Boolean structure of the formula. Then, we introduce branches and assignment strategies to formalize the notion of assignments in the symbolic case.

Boolean structure of a formula and branches. The transformation boolForm maps an α-TLA$^+$ expression to a Boolean formula over variables from $\{b_\ell \mid \ell \in \mathcal{L}\}$. The definition of boolForm can be found in Table 2. As $\text{boolForm}(\phi)$ is a formula in Boolean logic, a model of $\text{boolForm}(\phi)$ is a mapping from $\{b_\ell \mid \ell \in \mathcal{L}\}$ to $\mathbb{B} = \{\text{true}, \text{false}\}$. Take $S \subseteq \mathcal{L}$. The set S naturally defines a model induced by S, denoted $\mathcal{M}[S]$, by the requirement that $\mathcal{M}[S] \vDash b_\ell$ if and only if $\ell \in S$.

The boolForm transformation allows us to formulate the central notion of a branch: A set $Br \subseteq \mathcal{L}$ is called a *branch* of ϕ if the following constraints hold:

(a) The set Br induces a model of $\text{boolForm}(\phi)$, i.e., $\mathcal{M}[Br] \vDash \text{boolForm}(\phi)$, and
(b) The model $\mathcal{M}[Br]$ is minimal, that is, $\mathcal{M}[S] \nvDash \text{boolForm}(\phi)$ for every $S \subset Br$.

Table 2: The definition of boolForm(ϕ)

α-**TLA**$^+$ **expression** ϕ	boolForm(ϕ)
ℓ :: FALSE or ℓ :: $\star(v'_1, \ldots, v'_k)$ or ℓ :: $v' \in \phi_1$	b_ℓ
ℓ :: $\bigwedge_{i=1}^{s} \phi_i$	$\bigwedge_{i=1}^{s}$ boolForm(ϕ_i)
ℓ :: $\bigvee_{i=1}^{s} \phi_i$	$\bigvee_{i=1}^{s}$ boolForm(ϕ_i)
ℓ :: $\exists x \in \phi_1 \colon \ \phi_2$	boolForm(ϕ_2)
ℓ :: IF ϕ_1 THEN ϕ_2 ELSE ϕ_3	boolForm(ϕ_2) $\vee$ boolForm(ϕ_3)

Then, Branches(ϕ) is the set of all branches of ϕ.

Example 2. Let us look the α-TLA$^+$ expression ϕ given by

$$\ell_1 \ :: [[\ell_2 \ :: x' \in \star] \wedge [\ell_3 \ :: [[\ell_4 \ :: x' \in \star] \vee [\ell_5 \ :: x' \in \star]]]]$$

We know that boolForm(ϕ) $= b_{\ell_2} \wedge (b_{\ell_4} \vee b_{\ell_5})$. The set $S = \{\ell_2, \ell_4, \ell_5\}$ induces a model of boolForm(ϕ), but it is not a branch of ϕ because $\mathcal{M}[S]$ is not a minimal model. It is easy to see that ϕ has two branches $Br_1 = \{\ell_2, \ell_4\}$, and $Br_2 = \{\ell_2, \ell_5\}$. Therefore, we see that Branches(ϕ) $= \{Br_1, Br_2\}$. ◁

As our goal is to reason about the side-effects of variable assignments, the remainder of this section looks at how we can achieve this with the help of branches.

Assignment strategies. We want to statically mark some expressions as assignments, that is, pick a set $A \subseteq$ Labs(ϕ). Below, we formulate the critical properties we require from such a set, which we will later call an assignment strategy.

Most obviously, we want to consider only assignment candidates:

Definition 1. *A set $H \subseteq$ Labs(ϕ) is* homogeneous *if all the labels in H are assignment candidates. Formally, $H \subseteq$ cand(ϕ).*

If we choose an arbitrary homogeneous set H, it might lack assignments on some branches or have multiple assignments to the same variable on others. Formally, we say that H has a *covering index* $d \in \mathbb{N}_0$ if there is a branch $Br \in$ Branches(ϕ) and a variable $v' \in \mathit{Vars}'(\phi)$ for which $d = |Br \cap H \cap \text{cand}(v', \phi)|$. Now we define sets, that cover all branches with assignments:

Definition 2. *A homogeneous set C is a* covering *of ϕ, if it does not have* 0 *as a covering index. It is a* minimal covering *of ϕ, if it only has* 1 *as a covering index.*

Consider the TLA$^+$ formula $x' = y' \wedge y' = 2x'$. Its corresponding α-TLA$^+$ expression ℓ_0 :: $(\ell_1$ :: $x' \in \ell_2$ :: $\star(y') \wedge \ell_3$ :: $y' \in \ell_4$ :: $\star(x'))$ has a minimal covering $\{\ell_1, \ell_3\}$. However, there is no way to order the assignments to x' and y'. To detect such cases, we define acyclic sets:

Definition 3. *A homogeneous set A is* acyclic*, if there is a strict total order $\prec_A$ on A, with the following property: For every variable $v' \in V$, every branch $Br \in$ Branches and every pair of labels ℓ_i and ℓ_j in $A \cap Br$ the relation $\ell_i \lhd_{v'} \ell_j$ implies $\ell_i \prec_A \ell_j$.*

Having defined homogeneous, minimal covering, and acyclic sets, we can formulate the notion of an *assignment strategy.*

Definition 4. *Let ϕ be an α-TLA$^+$ expression. A set $A \subseteq \mathcal{L}$ is an assignment strategy for ϕ, if it is an acyclic minimal covering.*

Static assignment problem. Given an α-TLA$^+$ expression ϕ, our goal is to find an assignment strategy, or prove that none exists.

5 Finding Assignment Strategies with SMT

For a given α-TLA$^+$ expression ϕ, we construct an SMT formula $\theta(\phi)$, that encodes the properties of assignment strategies. Technically, $\theta(\phi)$ is defined as $\theta_H(\phi) \wedge \theta_C(\phi) \wedge \theta_A(\phi)$, and consists of:

1. A Boolean formula $\theta_H(\phi)$, that encodes homogeneity.
2. A Boolean formula $\theta_C(\phi)$, that encodes the minimal covering property.
3. A formula $\theta_A(\phi)$, that encodes acyclicity. This formula requires the theories of linear integer arithmetic and uninterpreted functions (*QF_UFLIA*).

In the following, Propositions 1, 3, and 4 formally establish the relation between ϕ and its three SMT counterparts. Together, the propositions allows us to prove the following theorem:

Theorem 1. *For every α-TLA$^+$ formula ϕ and $A \subseteq \text{Labs}(\phi)$, it holds that $\mathcal{M}[A] \vDash \theta(\phi)$ if and only if A is an assignment strategy for ϕ.*

5.1 Homogeneous Sets

We introduce a Boolean formula, whose models are exactly those induced by homogeneous sets. To this end, take the set of labels corresponding to expressions that are not assignment candidates, $\mathcal{N}(\phi)$, given by $\mathcal{N}(\phi) \coloneqq \text{Labs}(\phi) \setminus \text{cand}(\phi)$. Then, we define the following:

$$\theta_H(\phi) \coloneqq \bigwedge_{\ell \,\in\, \mathcal{N}(\phi)} \neg b_\ell$$

Proposition 1. *For every α-TLA$^+$ expression ϕ and $A \subseteq \text{Labs}(\phi)$, it holds that $\mathcal{M}[A] \vDash \theta_H(\phi)$ if and only if A is homogeneous.*

Table 3: The definition of $\delta_{v'}(\phi)$

α-TLA$^+$ expression ϕ	$\delta_{v'}(\phi)$
ℓ :: FALSE or ℓ :: $\star(v'_1, \ldots, v'_k)$	false
ℓ :: $w' \in \phi_1$	$\begin{cases} b_\ell & ; w' = v' \\ \text{false} & ; \text{otherwise} \end{cases}$
ℓ :: $\bigwedge_{i=1}^{s} \phi_i$	$\bigvee_{i=1}^{s} \delta_{v'}(\phi_i)$
ℓ :: $\bigvee_{i=1}^{s} \phi_i$	$\bigwedge_{i=1}^{s} \delta_{v'}(\phi_i)$
ℓ :: $\exists x \in \phi_1 : \phi_2$	$\delta_{v'}(\phi_2)$
ℓ :: IF ϕ_1 THEN ϕ_2 ELSE ϕ_3	$\delta_{v'}(\phi_2) \wedge \delta_{v'}(\phi_3)$

5.2 Minimal Covering Sets

Next we construct a Boolean formula $\theta^*_C(\phi)$, whose models are exactly those induced by covering sets. To this end, we define, for each $v' \in \mathit{Vars}'(\phi)$, the transformation $\delta_{v'}$ as shown in Table 3. Intuitively, $\delta_{v'}(\phi)$ is satisfiable exactly when there is an assignment to v' on every branch of ϕ. We then define

$$\theta^*_C(\phi) \coloneqq \bigwedge_{v' \in \mathit{Vars}'(\phi)} \delta_{v'}(\phi)$$

Formally, the following proposition holds:

Proposition 2. *For every α-TLA$^+$ expression ϕ and $A \subseteq \text{Labs}(\phi)$, it holds that $\mathcal{M}[A] \models \theta_H(\phi) \wedge \theta^*_C(\phi)$ if and only if A is a covering set for ϕ.*

It is easy to restrict coverings to the minimal coverings. To do this, we define the set of collocated labels, denoted Colloc(ϕ), as

$$\text{Colloc}(\phi) \coloneqq \{(\ell_1, \ell_2) \in \mathcal{L}^2 \mid \exists Br \in \text{Branches}(\phi) \,.\, \{\ell_1, \ell_2\} \subseteq Br\}$$

We can use this set to reason about minimal coverings: A minimal covering may contain, per variable, no more than one label from each pair of collocated assignments to that variable. We describe these labels by using the sets $\text{Colloc}_{v'}(\phi) \coloneqq \text{Colloc}(\phi) \cap \text{cand}(v', \phi)^2$ and

$$\text{Colloc}_{\mathit{Vars}'}(\phi) \coloneqq \bigcup_{v' \in \mathit{Vars}'(\phi)} \text{Colloc}_{v'}(\phi)$$

Then, the following SMT formula, in addition to $\theta^*_C(\phi)$, helps us find minimal covering sets:

$$\theta^{\exists!}(\phi) \coloneqq \bigwedge_{\substack{(i,j) \in \text{Colloc}_{\mathit{Vars}'}(\phi) \\ i<j}} \neg(b_i \wedge b_j)$$

We denote by $\theta_C(\phi)$ the formula $\theta^*_C(\phi) \wedge \theta^{\exists!}(\phi)$.

Proposition 3. *For every α-TLA$^+$ expression ϕ and $A \subseteq \text{Labs}(\phi)$, it holds that $\mathcal{M}[A] \models \theta_H(\phi) \wedge \theta_C(\phi)$ if and only if A is a minimal covering set for ϕ.*

5.3 Acyclic Assignments

The last step is reasoning about acyclicity. Recall that, for $\ell_1, \ell_2 \in \mathcal{L}$, the relation $\ell_1 \lhd_{v'} \ell_2$ holds if and only if $\ell_1 \in \text{cand}(v', \phi) \wedge v' \in \text{frozen}_\phi(\ell_2)$. It is prudent to see that $\lhd_{v'}$ is not, in general, a strict total order (possibly not even irreflexive). However, the acyclicity property states that we can find a strict total order, which agrees with all relations $\lhd_{v'}$, on all branches.

Take $\text{Colloc}_\lhd(\phi)$ to be the filtering of $\text{Colloc}(\phi)$ by the relations $\lhd_{v'}$, i.e. the set $\{(i,j) \in \text{Colloc}(\phi) \cap \text{cand}(\phi)^2 \mid \exists v' \in \mathit{Vars}'(\phi) \ . \ i \lhd_{v'} j\}$. The SMT formula describing acyclicity is as follows:

$$\theta^*_A(\phi) \coloneqq \bigwedge_{(i,j) \ \in \ \text{Colloc}_\lhd(\phi)} b_i \wedge b_j \Rightarrow R(i) < R(j)$$

where R is an uninterpreted $\mathcal{L} \to \mathbb{N}$ function, capturing assignment order. In practice, we take $\mathcal{L} = \mathbb{N}$. Unlike the previous formulas, $\theta^*_A(\phi)$ extends beyond Boolean logic, requiring both linear integer arithmetic and uninterpreted functions. Thus, a model for $\theta^*_A(\phi)$ is a pair (M, r), where M models the Boolean part of the formula, i.e. assigns truth values to each b_i, and $r \colon \mathbb{N} \to \mathbb{N}$ is the interpretation of R.

To simplify the analysis, we force R to be injective, when it is restricted to $\text{Labs}(\phi)$. Otherwise we could always construct an injective function from R, which respects the required inequalities. The formula we therefore consider is as follows:

$$\theta_A(\phi) \coloneqq \theta^*_A(\phi) \wedge \bigwedge_{\substack{\ell_1, \ell_j \ \in \ \text{Labs}(\phi) \\ \ell_i < \ell_j}} R(\ell_i) \neq R(\ell_j)$$

Proposition 4. *For every α-TLA$^+$ expression ϕ and $A \subseteq \text{Labs}(\phi)$, there is a function $r \colon \mathbb{N} \to \mathbb{N}$, for which $(\mathcal{M}[A], r) \vDash \theta_H(\phi) \wedge \theta_A(\phi)$ if and only if A is acyclic.*

6 Soundness of our Approach

In this section, we show the relation between assignment strategies and the original TLA$^+$ formulas. To this end, we introduce the notion of a slice. Together, branches allow us to rewrite a TLA$^+$ formula into an equivalent disjunction of slices.

In TLA$^+$, there are two kinds of variables: rigid variables that correspond to the variables declared with CONSTANT, and flexible variables whose values change during the course of an execution. Primed versions of the variables exist only for flexible variables and are used in transition formulas. Transition formulas in TLA$^+$ are first-order terms and formulas with flexible variables (unprimed and primed ones). We give the necessary definitions of TLA$^+$ semantics, whereas details can be found in [19]. An interpretation $\mathcal{I}$ defines a universe $|\mathcal{I}|$ of values and interprets each function symbol by a function and each predicate symbol by a relation. A state s is a mapping from unprimed flexible variables to values, and a state s' is a similar mapping for primed variables. A valuation ξ is a mapping from rigid variables to values. Given an interpretation $\mathcal{I}$, a pair of states (s, s'), and a valuation ξ, the semantics of a TLA$^+$ transition formula E is the standard predicate logic semantics of E with respect to the extended valuation of s, s', ξ. With these definitions, $M = (\mathcal{I}, \xi, s, s')$ is a model for E, if E is equivalent to true under M. Let ϕ be a formula and $S \subseteq \mathcal{L}$. We define ϕ *sliced by* S, denoted slice(ϕ, S) in Table 4.

Table 4: The definition of slice(ϕ, S)

α-TLA$^+$ formula ϕ	slice(ϕ, S)
ℓ :: FALSE	ℓ :: FALSE
ℓ :: $\star(v_1', \dots, v_1')$ or ℓ :: $v' \in \phi_1$	$\begin{cases} \phi & ; \ell \in S \\ \ell :: \text{FALSE} & ; \text{otherwise} \end{cases}$
ℓ :: $\bigwedge_{i=1}^{s} \phi_i$	ℓ :: $\bigwedge_{i=1}^{s}$ slice(ϕ_i, S)
ℓ :: $\bigvee_{i=1}^{s} \phi_i$	ℓ :: $\bigvee_{i=1}^{s}$ slice(ϕ_i, S)
$\ell :: \exists x \in \phi_1 : \phi_2$	$\ell :: \exists x \in \phi_1 :$ slice(ϕ_2, S)
ℓ :: IF ϕ_1 THEN ϕ_2 ELSE ϕ_3	ℓ :: IF ϕ_1 THEN slice(ϕ_2, S) ELSE slice(ϕ_3, S)

Below, we show that the branches and slices induced by them naturally decompose a TLA$^+$ formula. Let ϕ be an α-TLA$^+$ expression and $\gamma(\phi)$ its corresponding TLA$^+$ formula. Then, the following holds:

Proposition 5. *Let ϕ be an α-*TLA$^+$ *expression and $M = (\mathcal{I}, \xi, s, s')$ a model of the* TLA$^+$ *formula $\gamma(\phi)$. There exists a branch Br of ϕ such that M is also a model of γ (*slice*(ϕ, Br)).*

Proposition 6. *Let ϕ be an α-*TLA$^+$ *expression and $M = (\mathcal{I}, \xi, s, s')$ a model of the* TLA$^+$ *formula γ (*slice*(ϕ, Br)). Then, M is also a model of $\gamma(\phi)$.*

Proposition 7. *Let ϕ be an α-*TLA$^+$ *expression. For every S, $T \subseteq$* Labs(ϕ)*, every model M of the* TLA$^+$ *formula γ (*slice*(ϕ, S)), is also a model of γ (*slice*$(\phi, S \cup T)$).*

It is easy to see that Proposition 7 does not hold in the other direction. For instance, take the empty set as S and Labs(ϕ) as T. This implies the following:

$$\gamma(\text{slice}(\phi, S)) = \text{FALSE and slice}(\phi, S \cup T) = \phi.$$

Obviously, FALSE cannot have a model, regardless of whether $\gamma(\phi)$ has one or not.

Since Propositions 5 and 6 hold, it would suffice to consider the set Branches(ϕ), together with an assignment strategy, to generate symbolic transitions. However, it is often the case that, for two distinct branches Br_1 and Br_2, the same assignments in A are chosen, that is, the intersections $Br_1 \cap A$ and $Br_2 \cap A$ are the same. We show that one can reduce the number of considered symbolic transitions, by analyzing how various branches intersect A.

An assignment strategy A naturally defines an equivalence relation $\sim_A$ on Branches(ϕ), given by $Br_1 \sim_A Br_2$ if and only if $Br_1 \cap A = Br_2 \cap A$. We use the notation $[Br]_A$ to refer to the equivalence class of Br by $\sim_A$, that is, the set $\{X \in \text{Branches}(\phi) \mid Br \sim_A X\}$.

Definition 5. *Let ϕ be an α-TLA$^+$ expression, A an assignment strategy for ϕ and Br a branch of ϕ. Using $X = [Br]_A$ and $Y = \bigcup_{Z \in X} Z$, we define the symbolic transition generated by Br and A to be* $\mathrm{slice}(\phi, Y)$.

Example 3. Let us look Example 2 again. The formula ϕ has two assignment strategies $A_1 = \{\ell_2\}$, and $A_2 = \{\ell_4, \ell_5\}$. If the first assignment strategy A_1 is chosen, we have that $Br_1 \cap A_1 = Br_2 \cap A_1 = \{\ell_2\}$. This implies that Br_1 and Br_2 are in the same equivalence class, or $Br_1 \sim_{A_1} Br_2$. Therefore, we have only one symbolic transition which is exactly ϕ. However, if A_2 is selected, branches Br_1 and Br_2 are not equivalent because $Br_1 \cap A_2 = \{\ell_4\}$ and $Br_2 \cap A_2 = \{\ell_5\}$. Therefore, we have two symbolic transitions:

$$T_1 = \ell_1 :: [[\ell_2 :: x' \in \star] \wedge [\ell_3 :: [[\ell_4 :: x' \in \star] \vee \ell_5 :: \text{FALSE}]]]$$
$$T_2 = \ell_1 :: [[\ell_2 :: x' \in \star] \wedge [\ell_3 :: [\ell_4 :: \text{FALSE} \vee [\ell_5 :: x' \in \star]]]]$$

The first assignment strategy A_1 seems to be better than A_2 because A_1 generates fewer symbolic transitions than A_2. However, in this paper, we do not introduce any metric, by which we could compare assignment strategies. In the implementation, we use any strategy found by the SMT solver. ◁

The equivalence relation $\sim_A$ allows us to define a counterpart to Proposition 7:

Proposition 8. *Let ϕ be an α-TLA$^+$ expression. For any selection $Br_1, \dots, Br_k$ from the branches of ϕ, the following holds: If there exists a model M of the formula $\gamma(\mathrm{slice}(\phi, Br_1 \cup \dots \cup Br_k))$, then M must be a model of $\gamma(\mathrm{slice}(\phi, Br))$, for some branch $Br \in \mathrm{Branches}(\phi)$. Additionally, if there is an assignment strategy A for ϕ, such that $Br_1, \dots, Br_k$ all belong to the same equivalence class $[B]_A$, then M must be a model of $\gamma(\mathrm{slice}(\phi, Br))$, for some branch $Br \in [B]_A$.*

The following result allows us to use symbolic transitions, not individual branches:

Theorem 2. *Let ϕ be an α-TLA$^+$ expression and A an assignment strategy for ϕ. There is a model M of the TLA$^+$ formula $\gamma(\phi)$ if and only if there exists a $Br \in \mathrm{Branches}(\phi)$, such that M is a model of $\gamma(\psi)$, where ψ is the symbolic transition generated by Br and A.*

7 Preliminary Experiments and Potential Applications

Implementation and evaluation. Based on the theory presented in this paper, we have implemented a procedure to find assignment strategies and their corresponding symbolic transitions from TLA$^+$ specifications, or report that none exist. It uses Z3 as the background SMT solver.

We have chose specifications both from publicly available sources, e.g. EWD840 and Paxos from [1], and from a collection of algorithms we have encoded in TLA$^+$ ourselves. For each specification, we focus on the *Next* formula. We report the number of subexpressions in $\alpha(\mathit{Next})$, that is, $|\,\mathrm{Sub}(\alpha(\mathit{Next}))|$, the number of assignments in the strategy found by our procedure, the number of symbolic transitions

Table 5: Experimental results

Specification	**#subexpressions**	**size of strategy**	**#symbolic transitions**	**time (ms)**
aba [6]	86	48	8	271
nbacg [13]	126	82	13	205
EWD840 [11]	47	16	4	25
prodcons (Fig. 1)	12	4	2	19
Paxos [17]	60	16	4	29
nbac [25]	47	15	14	26
bcastFolklore [7]	41	17	4	28

computed and the total runtime. The results are presented in Table 5. Note that the results for the specification in Fig. 1 are as expected; all assignment candidates must be part of the strategy and we find two symbolic transitions corresponding to *Produce* and *Consume*. We also see that the number of symbolic transitions is generally much smaller than the number of transitions an explicit-state model checker would find, as even simple specifications, like in Figure 1, would generate numerous transitions in explicit state model checking, but only two symbolic transitions.

Applications. We illustrate an application of our technique for bounded model checking [4] by the means of the example in Figure 3. In this example, three processes pass a unique token in one direction, with the goal of computing the largest process identifier.

Our technique extracts three symbolic transitions T_1, T_2, and T_3, each T_i being equivalent to $P(i) \land id' = id$ for $1 \leq i \leq 3$. As common in bounded model checking, with $[\![F]\!]_{i,i+1}$ we denote the SMT encoding of a transition by action F from an ith to an $(i+1)$-th state. For instance, $[\![Next]\!]_{0,1}$ and $[\![T_3]\!]_{0,1}$ encode the transitions from the state 0 to the state 1 by *Next* and T_3. Likewise, $[\![Init]\!]_0$ encodes SMT constraints by *Init* on the initial states. One can use the SMT encodings introduced in [20,21].

MODULE *max*

EXTENDS *Naturals*
VARIABLE tok, max, id
$Init \triangleq tok = 1 \land id \in [1..3 \rightarrow Nat] \land max = 0$
$P(i) \triangleq tok = i \land tok' = 1 + i \,\%\, 3 \land max' = \text{IF } id[i] > max \text{ THEN } id[i] \text{ ELSE } max$
$Next \triangleq (P(1) \lor P(2) \lor P(3)) \land id' = id$

Fig. 3: A distributed maximum computation in a ring of three processes in TLA$^+$

$[\![Init]\!]_0 \quad [\![Next]\!]_{0,1} \quad [\![Next]\!]_{1,2} \quad [\![Next]\!]_{2,3}$

	$[\![T_1]\!]_{0,1}$	$[\![T_1]\!]_{1,2}$	$[\![T_1]\!]_{2,3}$
$[\![Init]\!]_0$	$[\![T_2]\!]_{0,1}$	$[\![T_2]\!]_{1,2}$	$[\![T_2]\!]_{2,3}$
	$[\![T_3]\!]_{0,1}$	$[\![T_3]\!]_{1,2}$	$[\![T_3]\!]_{2,3}$

Fig. 4: SMT formulas that are constructed when checking the executions up to length 4: using the action *Next* (left), and using symbolic transitions (right). The gray formulas are excluded from the SMT context during the exploration.

Figure 4 shows the SMT formulas that are constructed by a bounded model checker when exploring executions up to length 4. (For the sake of space, we omit the formulas that check property violation.) On one hand, the monolithic encoding that uses only *Next* has to keep all the formulas in the SMT context. On the other hand, by incrementally checking satisfiability of the SMT context, the model checker can discover that some formulas — for example, $[\![T_2]\!]_{0,1}$ and $[\![T_3]\!]_{1,2}$ — lead to unsatisfiability and prune them from the SMT context. Similar approach improves efficiency of bounded model checking `C` programs [5][Ch. 16], hence, we expect it to be effective for the verification of TLA$^+$ specifications too.

8 Conclusions

We have introduced a technique to compute symbolic transitions of a TLA$^+$ specification by finding expressions that can be interpreted as assignments. Importantly, we designed the technique with soundness in mind. Detailed proofs can be found in the report [16]. We believe that our results can be used as a first preprocessing step when developing a symbolic model checker or a type checker for TLA$^+$.

As in the case of TLC, one can find TLA$^+$ specifications, for which no assignment strategy exists. However, TLA$^+$ users are systematically checking their specifications with TLC, in order to find simple errors. Hence, most of the benchmarks already come in a form compatible with TLC. Thus, we expect our approach to also work in practice. Based on these ideas, we are currently developing a symbolic model checker for TLA$^+$.

Acknowledgments. We are grateful to Stephan Merz for insightful discussions on semantics of TLA$^+$. We thank anonymous reviewers for their helpful suggestions.

References

1. A collection of TLA$^+$ specifications. `https://github.com/tlaplus/Examples/`, [Online; accessed 21-October-2017]
2. Azmy, N., Merz, S., Weidenbach, C.: A rigorous correctness proof for pastry. In: Abstract State Machines, Alloy, B, TLA, VDM, and Z. pp. 86–101. Springer (2016)
3. Beyer, D., Keremoglu, M.E.: CPAchecker: A tool for configurable software verification. In: CAV. pp. 184–190 (2011)

4. Biere, A., Cimatti, A., Clarke, E.M., Zhu, Y.: Symbolic model checking without BDDs. In: TACAS. LNCS, vol. 1579, pp. 193–207 (1999)
5. Biere, A., Heule, M., van Maaren, H.: Handbook of satisfiability, vol. 185. IOS press (2009)
6. Bracha, G., Toueg, S.: Asynchronous consensus and broadcast protocols. Journal of the ACM 32(4), 824–840 (1985)
7. Chandra, T.D., Toueg, S.: Unreliable failure detectors for reliable distributed systems. Journal of the ACM 43(2), 225–267 (1996)
8. Chaudhuri, K., Doligez, D., Lamport, L., Merz, S.: The TLA$^+$ proof system: Building a heterogeneous verification platform. In: Theoretical aspects of computing. pp. 44–44. Springer-Verlag (2010)
9. Clarke, E., Grumberg, O., Jha, S., Lu, Y., Veith, H.: Counterexample-guided abstraction refinement for symbolic model checking. J. ACM 50(5), 752–794 (2003)
10. Conchon, S., Goel, A., Krstic, S., Mebsout, A., Zaïdi, F.: Cubicle: A parallel smt-based model checker for parameterized systems - tool paper. In: CAV. pp. 718–724 (2012)
11. Dijkstra, E.W., Feijen, W.H., Van Gasteren, A.M.: Derivation of a termination detection algorithm for distributed computations. In: Control Flow and Data Flow: concepts of distributed programming, pp. 507–512. Springer (1986)
12. Gafni, E., Lamport, L.: Disk paxos. Distributed Computing 16(1), 1–20 (2003)
13. Guerraoui, R.: On the hardness of failure-sensitive agreement problems. Information Processing Letters 79(2), 99–104 (2001)
14. Konnov, I., Lazić, M., Veith, H., Widder, J.: A short counterexample property for safety and liveness verification of fault-tolerant distributed algorithms. In: POPL. pp. 719–734 (2017)
15. Kroening, D., Tautschnig, M.: CBMC - C bounded model checker - (competition contribution). In: TACAS. pp. 389–391 (2014)
16. Kukovec, J., Tran, T.H., Konnov, I.: Extracting symbolic transitions from TLA$^+$ specifications (technical report 2018). `http://forsyte.at/wp-content/uploads/abz2018_full.pdf`, [Online; accessed 7-Feb-2018]
17. Lamport, L.: The part-time parliament. ACM TCS 16(2), 133–169 (1998)
18. Lamport, L.: Specifying systems: the TLA$^+$ language and tools for hardware and software engineers. Addison-Wesley Longman Publishing Co., Inc. (2002)
19. Merz, S.: The specification language TLA$^+$. In: Logics of specification languages, pp. 401–451. Springer (2008)
20. Merz, S., Vanzetto, H.: Automatic verification of TLA$^+$ proof obligations with SMT solvers. In: LPAR. vol. 7180, pp. 289–303. Springer (2012)
21. Merz, S., Vanzetto, H.: Harnessing SMT solvers for TLA+ proofs. ECEASST 53 (2012)
22. Moraru, I., Andersen, D.G., Kaminsky, M.: There is more consensus in egalitarian parliaments. In: SOSP. pp. 358–372. ACM (2013)
23. Newcombe, C., Rath, T., Zhang, F., Munteanu, B., Brooker, M., Deardeuff, M.: How Amazon web services uses formal methods. Comm. ACM 58(4), 66–73 (2015)
24. Ongaro, D.: Consensus: Bridging theory and practice. Ph.D. thesis, Stanford U. (2014)
25. Raynal, M.: A case study of agreement problems in distributed systems: Non-blocking atomic commitment. In: HASE. pp. 209–214 (1997)
26. Yu, Y., Manolios, P., Lamport, L.: Model checking TLA$^+$ specifications. In: Correct Hardware Design and Verification Methods, pp. 54–66. Springer (1999)

A Extended Definitions

Table 8 gives us a full formal definition of the function depth which maps an expression ϕ in our α-TLA$^+$ abstract syntax to a natural number. Proofs of Propositions 5 and 6 are based on induction on the depth of an expression ϕ in the α-TLA$^+$ abstract syntax.

Table 6: The definition of Sub(ϕ)

α-TLA$^+$ expression ϕ	Sub(ϕ)
ℓ :: FALSE or ℓ :: $\star(v'_1, \ldots, v'_k)$	$\{\phi\}$
ℓ :: $v' \in \phi_1$	$\{\phi, \phi_1\}$
ℓ :: $\bigwedge_{i=1}^{s} \phi_i$ or ℓ :: $\bigvee_{i=1}^{s} \phi_i$	$\{\phi\} \cup \bigcup_{i=1}^{s} \mathrm{Sub}(\phi_i)$
ℓ :: $\exists x \in \phi_1 : \ \phi_2$	$\{\phi\} \cup \mathrm{Sub}(\phi_1) \cup \mathrm{Sub}(\phi_2)$
ℓ :: IF ϕ_1 THEN ϕ_2 ELSE ϕ_3	$\{\phi\} \cup \mathrm{Sub}(\phi_1) \cup \mathrm{Sub}(\phi_2) \cup \mathrm{Sub}(\phi_3)$

Table 7: The definition of cand(v', ϕ)

α-TLA$^+$ expression ϕ	cand(v', ϕ)
ℓ :: FALSE or ℓ :: $\star(v'_1, \ldots, v'_k)$	$\emptyset$
ℓ :: $w' \in \phi_1$	$\begin{cases} \{\ell\} & ; w' = v' \\ \emptyset & ; \text{otherwise} \end{cases}$
ℓ :: $\bigwedge_{i=1}^{s} \phi_i$ or ℓ :: $\bigvee_{i=1}^{s} \phi_i$	$\bigcup_{i=1}^{s} \mathrm{cand}(v', \phi_i)$
ℓ :: $\exists x \in \phi_1 : \ \phi_2$	$\mathrm{cand}(v', \phi_2)$
ℓ :: IF ϕ_1 THEN ϕ_2 ELSE ϕ_3	$\mathrm{cand}(v', \phi_2) \cup \mathrm{cand}(v', \phi_3)$

B Detailed Proofs

Our propositions require additional lemmas, which we introduce only in the appendix. Specifically:

- Proposition 2 requires Lemmas 2, 9, 10, 8, 5, 6, and 7.
- Proposition 4 requires Lemma 11.
- Proposition 5 requires Lemmas 5, 4, 3, 6, and 7.
- Proposition 6 requires Lemmas 5, 3, 4, 6, and 7.
- Proposition 8 requires Lemmas 4, 3, 5, 6, and 7, .

Table 8: The definition of depth

$\text{depth}(\phi) = 0$	if $\phi = \ell :: v' \in ex_\alpha$, or $\phi = \ell :: \text{FALSE}$, or $\phi = \ell :: \star(v', \ldots, v')$
$\text{depth}(\phi) = 1 + \max(\text{depth}(\phi_1), \text{depth}(\phi_2))$	if $\phi = \ell :: \phi_1 \lor \phi_2$, or $\phi = \ell :: \phi_1 \land \phi_2$
$\text{depth}(\phi) = 1 + \text{depth}(\phi_1)$	if $\phi = \ell :: \exists x \in S \,.\, \phi_1$
$\text{depth}(\phi) = 1 + \max(\text{depth}(\phi_2), \text{depth}(\phi_3))$	if $\phi = \ell :: \text{IF } \phi_1 \text{ THEN } \phi_2$ $\text{ELSE } \phi_3$

B.1 Additional Lemmas

Lemma 1. *Let ϕ be an α-TLA$^+$ expression. For any set $A \subseteq \mathcal{L}$, it holds that*

$$\mathcal{M}[A] \vDash \text{boolForm}(\phi) \iff \mathcal{M}[A \cap \text{Labs}(\phi)] \vDash \text{boolForm}(\phi)$$

Proof. We prove this by induction on the structure of ϕ:

- $\phi = \ell :: \text{FALSE}$: By definition, $\text{boolForm}(\phi) = b_\ell$ and $\text{Labs}(\phi) = \{\ell\}$. It is clear that $\ell \in A \iff \ell \in A \cap \{\ell\}$. If we look at the definition of the induced model, we can conclude the following:

$$\mathcal{M}[A] \vDash b_\ell \iff \ell \in A \iff \ell \in A \cap \{\ell\} \iff \mathcal{M}[A \cap \{\ell\}] \vDash b_\ell$$

- $\phi = \ell :: \star(v'_1, \ldots, v'_k)$: Same as for $\phi = \ell :: \text{FALSE}$.
- $\phi = \ell :: v' \in \hat{\ell} :: \star(v'_1, \ldots, v'_k)$: By definition, $\text{Labs}(\phi) = \{\ell, \hat{\ell}\}$. Again, $\ell \in A \iff \ell \in A \cap \text{Labs}(\phi)$, the rest is the same as for $\phi = \ell :: \text{FALSE}$.
- $\phi = \ell :: \bigwedge_{i=1}^{s} \phi_i$: Assume as the induction hypothesis, that the lemma holds for every $\phi_i, i \in \{1, \ldots, s\}$. As $\text{boolForm}(\phi) = \bigwedge_{i=1}^{s} \text{boolForm}(\phi_i)$ by definition, we know that

$$\mathcal{M}[A] \vDash \text{boolForm}(\phi) \iff \mathcal{M}[A] \vDash \text{boolForm}(\phi_i), \text{ for all } i \in \{1, \ldots, s\}$$

Take an arbitrary $i \in \{1, \ldots, s\}$. By the induction hypothesis

$$\mathcal{M}[A] \vDash \text{boolForm}(\phi_i) \iff \mathcal{M}[A \cap \text{Labs}(\phi_i)] \vDash \text{boolForm}(\phi_i)$$

By applying the hypothesis again, it is also the case that

$$\mathcal{M}[A \cap \text{Labs}(\phi)] \vDash \text{boolForm}(\phi_i) \iff \mathcal{M}[(A \cap \text{Labs}(\phi)) \cap \text{Labs}(\phi_i)] \vDash \text{boolForm}(\phi_i)$$

Since $\text{Labs}(\phi) \cap \text{Labs}(\phi_i) = \text{Labs}(\phi_i)$ we can conclude that

$$\mathcal{M}[A] \vDash \text{boolForm}(\phi_i) \iff \mathcal{M}[A \cap \text{Labs}(\phi)] \vDash \text{boolForm}(\phi_i)$$

Since i is arbitrary, this holds for every ϕ_i, so the lemma holds for such ϕ.

– $\phi = \ell :: \bigvee_{i=1}^{s} \phi_i$: Analogous to the case where $\phi = \ell :: \bigwedge_{i=1}^{s} \phi_i$.
– $\phi = \ell :: \exists x \in \psi \colon \phi_0$: Assume as the induction hypothesis, that the lemma holds for ϕ_0. As $\text{boolForm}(\phi) = \text{boolForm}(\phi_0)$ by definition, we know

$$\begin{aligned}\mathcal{M}[A] \models \text{boolForm}(\phi) &\iff \mathcal{M}[A] \models \text{boolForm}(\phi_0)\\ &\iff \mathcal{M}[A \cap \text{Labs}(\phi_0)] \models \text{boolForm}(\phi)\end{aligned}$$

and

$$\mathcal{M}[A \cap \text{Labs}(\phi)] \models \text{boolForm}(\phi_0) \iff \mathcal{M}[A \cap \text{Labs}(\phi) \cap \text{Labs}(\phi_0)] \models \text{boolForm}(\phi_0)$$

Since $\text{Labs}(\phi_0) \subseteq \text{Labs}(\phi)$ we know that for any A the sets $A \cap \text{Labs}(\phi_0)$ and $A \cap \text{Labs}(\phi) \cap \text{Labs}(\phi_0)$ are the same, thus the lemma holds.
– $\phi = \ell :: \text{IF } \phi_1 \text{ THEN } \phi_2 \text{ ELSE } \phi_3$: Analogous to the case where $\phi = \ell :: \phi_2 \vee \phi_3$ as $\text{boolForm}(\phi) = \text{boolForm}(\phi_2) \vee \text{boolForm}(\phi_3)$ and $\text{Labs}(\phi_2) \cup \text{Labs}(\phi_3) \subseteq \text{Labs}(\phi)$.

Thus the lemma holds for any α-TLA$^+$ expression ϕ. □

Lemma 2. *Let ϕ be an α-TLA$^+$ expression. For any set $A \subseteq \mathcal{L}$ and any variable $v' \in Vars'(\phi)$, it holds that*

$$\mathcal{M}[A] \models \delta_{v'}(\phi) \iff \mathcal{M}[A \cap \text{cand}(v', \phi)] \models \delta_{v'}(\phi)$$

Proof. Analogous to the proof of Lemma 1. □

Lemma 3. *Let ϕ be an α-TLA$^+$ expression. For any set $A \subseteq \mathcal{L}$, it holds that*

$$\text{slice}(\phi, A) = \text{slice}(\phi, A \cap \text{Labs}(\phi))$$

Proof. Analogous to the proof of Lemma 1. □

Lemma 4. *Let ϕ be an α-TLA$^+$ expression. If ϕ has the shape $\phi = \ell :: \bigwedge_{i=1}^{s} \phi_i$ it follows that every branch of ϕ is a union of branches for each ϕ_i and vice-versa. Formally:*

$$\text{Branches}(\phi) = \left\{ \bigcup_{i=1}^{s} Br_i \mid \forall i \in \{1, \ldots, s\} \;.\; Br_i \in \text{Branches}(\phi_i) \right\}$$

Proof. Take an arbitrary $Br \in \text{Branches}(\phi)$. By the definition of a branch, $\mathcal{M}[Br] \models \text{boolForm}(\phi)$. We define $Br_i := Br \cap \text{Labs}(\phi_i)$ for each $i = 1, \ldots, s$. Then, $B = \bigcup_{i=1}^{s} Br_i$ by construction, since $\text{Labs}(\phi) = \bigcup_{i=1}^{s} \text{Labs}(\phi_i)$. Because each subexpression of ϕ has a unique label, the sets $\text{Labs}(\phi_i)$ are pairwise disjoint. Take an arbitrary $i \in \{1, \ldots, s\}$. Since $\text{boolForm}(\phi)$ implies $\text{boolForm}(\phi_i)$, we know $\mathcal{M}[Br] \models \text{boolForm}(\phi_i)$. By Lemma 1, it must be the case that $\mathcal{M}[Br_i] \models \text{boolForm}(\phi_i)$ as well. Now take an arbitrary nonempty $T \subseteq Br_i$. Because Br induces a minimal model, we know $\mathcal{M}[Br \setminus T] \not\models \text{boolForm}(\phi)$. If we look at any $j \neq i$, since $\text{Labs}(\phi_i)$ and $\text{Labs}(\phi_j)$ are disjoint, the set $(Br \setminus T) \cap \text{Labs}(\psi_j)$ is just $Br \cap \text{Labs}(\phi_j) = Br_j$.

Clearly, $\mathcal{M}[Br \setminus T] \vDash \text{boolForm}(\phi_j)$, by Lemma 1, for all $j \neq i$, so the only reason $\mathcal{M}[Br \setminus T]$ does not model $\text{boolForm}(\phi)$ is because $\mathcal{M}[Br \setminus T] \nvDash \text{boolForm}(\phi_i)$ Since $(Br \setminus T) \cap \text{Labs}(\phi_i)$ is $Br_i \setminus T$, which is a proper subset of Br_i for every nonempty subset T of Br_i, we can conclude that $\mathcal{M}[S] \nvDash \text{boolForm}(\phi)$ must hold for every $S \subseteq Br_i$, which proves that Br_i is indeed a branch of ϕ_i, for every $i \in \{1, \ldots, s\}$.

Alternatively, take arbitrary branches $Br_1, \ldots, Br_s$ of subexpressions, such that $Br_1 \in \text{Branches}(\phi_1), \ldots, Br_s \in \text{Branches}(\phi_s)$. Define $Br := \bigcup_{i=1}^{s} Br_i$. We must show that this Br is a branch of ϕ. Take an arbitrary $i \in \{1, \ldots, s\}$. By definition, $\mathcal{M}[Br_i] \vDash \text{boolForm}(\phi_i)$. Lemma 1 tells us that $\mathcal{M}[Br] \vDash \text{boolForm}(\phi_i)$ exactly when $\mathcal{M}[Br \cap \text{Labs}(\phi_i)] \vDash \text{boolForm}(\phi_i)$. Because Br_i is minimal, it must be the case that $Br_i \cap \text{Labs}(\phi_i)$ equals Br_i. If it were some proper subset, $S \subset Br_i$, applying Lemma 1 to Br_i would give us

$$\mathcal{M}[Br_i] \vDash \text{boolForm}(\phi_i) \iff \mathcal{M}[S] \vDash \text{boolForm}(\phi_i)$$

which contradicts the property that for every $T \subset Br_i$ we know $\mathcal{M}[T] \nvDash \text{boolForm}(\phi_i)$. It remains to be seen that $Br \cap \text{Labs}(\phi_i) = Br_i$. Expanding Br tells us

$$Br \cap \text{Labs}(\phi_i) = \bigcup_{j=1}^{s} Br_j \cap \text{Labs}(\phi_i)$$

If $i \neq j$ then, as $Br_j \subseteq \text{Labs}(\phi_j)$ and the label sets $\text{Labs}(\phi_i)$ and $\text{Labs}(\phi_j)$ are disjoint, we conclude $\text{Labs}(\phi_i) \cap Br_j = \emptyset$. So $\mathcal{M}[Br] \vDash \text{boolForm}(\phi_i)$. As i was arbitrary, this means $\mathcal{M}[Br] \vDash \bigwedge_{i=1}^{s} \text{boolForm}(\phi_i)$. To see that Br induces a minimal model, take an arbitrary nonempty $T \subseteq Br$. Then, $S := Br \setminus T$ is a proper subset of Br. There must exist an i, for which $T \cap Br_i \neq \emptyset$. By Lemma 1, we know that

$$\begin{aligned} \mathcal{M}[S] \vDash \text{boolForm}(\phi_i) &\iff \mathcal{M}[S \cap \text{Labs}(\phi_i)] \vDash \text{boolForm}(\phi_i) \\ &\iff \mathcal{M}[(Br \setminus T) \cap \text{Labs}(\phi_i)] \vDash \text{boolForm}(\phi_i) \\ &\iff \mathcal{M}[Br_i \setminus T] \vDash \text{boolForm}(\phi_i) \end{aligned}$$

But Br_i is a branch and $Br_i \setminus T$ is its proper subset, so $\mathcal{M}[Br_i \setminus T] \nvDash \text{boolForm}(\phi_i)$ and consequently, $\mathcal{M}[S] \nvDash \text{boolForm}(\phi)$, for any proper subset $S \subset Br$. Therefore, Br is a branch of ϕ. □

Lemma 5. *Let ϕ be an α-TLA$^+$ expression. If ϕ has the shape $\phi = \ell :: \bigvee_{i=1}^{s} \phi_i$ it follows that every branch of ϕ is a branch of some ϕ_i and vice-versa. Formally:*

$$\text{Branches}(\phi) = \bigcup_{i=1}^{s} \text{Branches}(\phi_i)$$

Proof. Take an arbitrary $i \in \{1, \ldots, s\}$ and $Br \in \text{Branches}(\phi_i)$. Since $\mathcal{M}[Br] \vDash \text{boolForm}(\phi_i)$ it follows that $\mathcal{M}[Br] \vDash \bigvee_{j=1}^{s} \text{boolForm}(\phi_j)$. To see that Br is minimal, take an arbitrary $S \subset Br$. By definition, $\mathcal{M}[S] \nvDash \text{boolForm}(\phi_i)$. To see that it cannot induce a model for $\text{boolForm}(\phi_j)$, where $i \neq j$, we note that $\text{Labs}(\phi_i) \cap \text{Labs}(\phi_j) = \emptyset$ and, by Lemma 1,

$$\mathcal{M}[S] \vDash \text{boolForm}(\phi_j) \iff \mathcal{M}[S \cap \text{Labs}(\phi_j)] \vDash \text{boolForm}(\phi_j)$$

Since $S \subset \text{Labs}(\phi_i)$ we know that $S \cap \text{Labs}(\phi_j) = \emptyset$. As no boolForm formula is a tautology, by construction, it follows that $\mathcal{M}[\emptyset]$ cannot model boolForm(ϕ_j) for $j \neq i$. So S cannot induce a model for $\bigvee_{j=1}^{s} \text{boolForm}(\phi_j)$ and thus Br is a branch of ϕ.

Alternatively, take a $Br \in$ Branches(ϕ). There must exist some $i \in \{1, \ldots, s\}$ for which $\mathcal{M}[Br] \vDash \text{boolForm}(\phi_i)$. We show that $Br \cap \text{Labs}(\phi_j) = \emptyset$ for all $i \neq j$ by contradiction: Assume that for some $j \neq i$ there exists a $x \in \text{Labs}(\phi_j) \cap Br$. It is always the case that Labs(ϕ_i) and Labs(ϕ_j) are disjoint. If we invoke Lemma 1, we see that

$$\mathcal{M}[Br] \vDash \text{boolForm}(\phi_i) \iff \mathcal{M}[Br \cap \text{Labs}(\phi_i)] \vDash \text{boolForm}(\phi_i)$$

Because x belongs to Labs(ϕ_j) it must be the case that $Br \setminus \{x\}$ also induces a model for boolForm(ϕ_i). But this is a contradiction, since Br is a branch and $Br \setminus \{x\}$ is a proper subset. Consequently, the assumption is false and $Br \cap \text{Labs}(\phi_j) = \emptyset$ for all $i \neq j$. It remains to see that no $S \subset Br$ can induce a model for boolForm(ϕ_i). Take an arbitrary $S \subset Br$. Since, for $i \neq j$, $Br \cap \text{Labs}(\phi_j) = \emptyset$ then $\mathcal{M}[Br] \nvDash \text{boolForm}(\phi_j)$. Because $\mathcal{M}[S] \nvDash \text{boolForm}(\phi)$, as Br is a branch, we must conclude that $\mathcal{M}[S] \nvDash \text{boolForm}(\phi_i)$. But that means Br is a branch of ϕ_i. □

Lemma 6. *Let ϕ be an α-TLA$^+$ expression. If ϕ has the shape $\phi = \ell :: \exists x \in \psi \, . \, \phi_0$ it follows that branches of ϕ are exactly branches of ϕ_0. Formally:*

$$\text{Branches}(\phi) = \text{Branches}(\phi_0)$$

Proof. Clearly, as boolForm(ϕ) = boolForm(ϕ_0) by definition, we know

$$\mathcal{M}[T] \vDash \text{boolForm}(\phi) \iff \mathcal{M}[T] \vDash \text{boolForm}(\phi_0)$$

for any $T \subseteq \mathcal{L}$, in particular also for branches. □

Lemma 7. *Let ϕ be an α-TLA$^+$ expression. If $\phi = \ell ::$ IF ϕ_1 THEN ϕ_2 ELSE ϕ_3 it follows that every branch of ϕ is a branch of either ϕ_2 or ϕ_3 and vice-versa. Formally:*

$$\text{Branches}(\phi) = \text{Branches}(\phi_2) \cup \text{Branches}(\phi_3)$$

Proof. Analogous to the proof of Lemma 5, since boolForm(ϕ) = boolForm(ϕ_2) $\vee$ boolForm(ϕ_3). □

Lemma 8. *Let $\phi = \ell :: \bigwedge_{i=1}^{s} \phi_i$ be an α-TLA$^+$ expression and J a set that intersects every branch of ϕ. Then, J intersects every branch of some ϕ_i non-trivially as well. Formally, take a set $J \subseteq \mathcal{L}$ with the property that*

$$\forall Br \in \text{Branches}(\phi) \, . \, J \cap Br \neq \emptyset$$

Then, the following holds:

$$\exists k \in \{1, \ldots, s\} \, . \, \forall Br \in \text{Branches}(\phi_k) \, . \, J \cap Br \neq \emptyset$$

Proof. We prove this by contradiction. Assume that for every $k \in \{1, \dots, s\}$ we can find a $Br_k \in \text{Branches}(\phi_k)$ for which $J \cap B_k = \emptyset$. If we take $Br := \bigcup_{k=1}^{s} Br_k$, we generate a branch of ϕ, by Lemma 4. Then, by assumption, $J \cap B \neq \emptyset$. However, from the way we have defined Br, we see that

$$J \cap Br = J \cap \bigcup_{k=1}^{s} Br_k = \bigcup_{k=1}^{s} (J \cap Br_k) = \bigcup_{k=1}^{s} \emptyset = \emptyset$$

From this contradiction, we deduce that the lemma must hold. □

Lemma 9. *Let ϕ be and α-TLA$^+$ expression. For any $v' \in Vars'(\phi)$ and $S \subseteq \mathcal{L}$, it holds that if $\mathcal{M}[S] \vDash \delta_{v'}(\phi)$ then $\mathcal{M}[\mathcal{L} \setminus S] \vDash \neg\,\text{boolForm}(\phi)$.*

Proof. We will use induction on the structure of ϕ:

- $\phi = \ell$ :: FALSE : Since $\delta_{v'}(\phi) = \text{false}$ the implication is vacuously true as no model exists.
- $\phi = \ell$ :: $\star\,(v'_1, \dots, v'_k)$: Same as for $\phi = \ell$ :: FALSE.
- $\phi = \ell$ :: $w' \in \psi$: If $\delta_{v'}(\phi) = \text{false}$ the implication is vacuously true, since no model exists. If $\delta_{v'}(\phi) = b_\ell$ then $\neg\,\text{boolForm}(\phi) = \neg b_\ell$ and

  $$\mathcal{M}[S] \vDash b_\ell \iff \ell \in S \iff \ell \notin \mathcal{L} \setminus S \iff \mathcal{M}[\mathcal{L} \setminus S] \vDash \neg b_\ell$$

  Thus, the implication holds.
- $\phi = \ell$ :: $\bigwedge_{i=1}^{s} \phi_i$: Assume as the induction hypothesis, that the lemma holds for each ϕ_i. Let $\mathcal{M}[S] \vDash \delta_{v'}(\phi)$. By definition, $\delta_{v'}(\phi) = \bigvee_{i=1}^{s} \delta_{v'}(\phi_i)$, so we know that there exists a $j \in \{1, \dots, s\}$, for which $\mathcal{M}[S] \vDash \delta_{v'}(\phi_j)$. By the induction hypothesis, we then know $\mathcal{M}[\mathcal{L} \setminus S] \vDash \neg\,\text{boolForm}(\phi_j)$. Since $\neg\,\text{boolForm}(\phi) = \bigvee_{i=1}^{s} \neg\,\text{boolForm}(\phi_i)$ it also follows that $\mathcal{M}[\mathcal{L} \setminus S] \vDash \neg\,\text{boolForm}(\phi)$, as required.
- $\phi = \ell$ :: $\bigvee_{i=1}^{s} \phi_i$: Analogous to the previous case.
- $\phi = \ell$ :: $\exists x \in \psi$: ϕ_0: Assume the lemma holds for ϕ_0. It is obvious that, since $\delta_{v'}(\phi) = \delta_{v'}(\phi_0)$ and $\text{boolForm}(\phi) = \text{boolForm}(\phi_0)$, the lemma holds for ϕ as well.
- $\phi = \ell$ :: IF ϕ_1 THEN ϕ_2 ELSE ϕ_3 : Analogous to the disjunction case, since $\text{boolForm}(\phi) = \text{boolForm}(\phi_2 \vee \phi_3)$ and $\delta_{v'}(\phi) = \delta_{v'}(\phi_2 \vee \phi_3)$.

Thus the lemma holds for any α-TLA$^+$ expression ϕ. □

Lemma 10. *If ψ is a Boolean formula in NNF with only negated atoms $\neg b_{\ell_1}, \dots, \neg b_{\ell_k}$ and $S \subseteq \mathcal{L}$ is a set for which $\mathcal{M}[S] \vDash \psi$ then $\mathcal{M}[J] \vDash \psi$, for every $J \subseteq S$.*

Proof. We can prove this by induction on the structure of ψ:

- $\psi = \neg b_\ell$: By definition,

  $$\mathcal{M}[S] \vDash \neg b_\ell \iff \ell \notin S$$

  Since $J \subseteq S$, we know $\ell \notin S$ implies $\ell \notin J$. Thus, $\mathcal{M}[J] \vDash b_\ell$.

– $\psi = \bigwedge_{i=1}^{s} \psi_i$: Assume as the induction hypothesis, that the lemma holds for all ψ_i. We know

$$\mathcal{M}[S] \vDash \psi \iff \mathcal{M}[S] \vDash \psi_i, \text{ for all } i \in \{1, \ldots, s\}$$

If $\mathcal{M}[S] \vDash \psi$ and $J \subseteq S$ it follows from the induction hypothesis, that $\mathcal{M}[J] \vDash \psi_i$ for all i. So clearly, $\mathcal{M}[J] \vDash \psi$.

– $\psi = \bigvee_{i=1}^{s} \psi_i$: Assume as the induction hypothesis, that the lemma holds for all ψ_i. We know

$$\mathcal{M}[S] \vDash \psi \iff \mathcal{M}[S] \vDash \psi_i, \text{ for some } i \in \{1, \ldots, s\}$$

If $\mathcal{M}[S] \vDash \psi$ and $J \subseteq S$ it follows from the induction hypothesis, that $\mathcal{M}[J] \vDash \psi_i$ for some i. So clearly, $\mathcal{M}[J] \vDash \psi$.

We conclude that the lemma holds for any propositional formula ψ in NNF.

□

Lemma 11. *If $<$ is a strict total order on Y and $f\colon X \to Y$ is injective then $\prec$ defined by*

$$x_1 \prec x_2 \iff f(x_1) < f(x_2)$$

is a strict total order on X.

Proof. We need to show transitivity, asymmetry, irreflexivity and totality of the relation $\prec$.

transitivity:

$$\begin{aligned} x_1 \prec x_2 \land x_2 \prec x_3 &\iff f(x_1) < f(x_2) \land f(x_2) < f(x_3) \\ &\implies f(x_1) < f(x_3) \\ &\iff x_1 \prec x_3 \end{aligned}$$

asymmetry:

$$\begin{aligned} x_1 \prec x_2 &\iff f(x_1) < f(x_2) \\ &\implies \neg(f(x_2) < f(x_1)) \\ &\iff \neg(x_2 \prec x_1) \end{aligned}$$

irreflexivity:

$$\begin{aligned} \forall y \in Y \,.\, \neg(y < y) &\implies \forall x \in X \,.\, \neg(f(x) < f(x)) \\ &\iff \forall x \in X \,.\, \neg(x \prec x) \end{aligned}$$

totality:

$$\forall y_1, y_2 \in Y \,.\, y_1 < y_2 \lor y_2 < y_1 \lor y_1 = y_2$$

implies

$$\forall x_1, x_2 \in X \ . \ f(x_1) < f(x_2) \lor f(x_2) < f(x_1) \lor f(x_1) = f(x_2)$$

which is equivalent to

$$\forall x_1, x_2 \in X \ . \ x_1 \prec x_2 \lor x_2 \prec x_1 \lor x_1 = x_2$$

for injective f.

Thus $\prec$ is a strict total order on X ☐

B.2 Proofs of Section 5

Proposition 1. *For every* α-TLA$^+$ *expression* ϕ *and* $A \subseteq \text{Labs}(\phi)$*, it holds that* $\mathcal{M}[A] \vDash \theta_H(\phi)$ *if and only if* A *is homogeneous.*

Proof. Firstly, assume $\mathcal{M}[A] \vDash \theta_H(\phi)$. Take an arbitrary $\ell \in \mathcal{N}(\phi)$. Then

$$\mathcal{M}[A] \vDash \theta_H(\phi) \Rightarrow \mathcal{M}[A] \vDash \neg b_\ell \iff \ell \notin A$$

So every element of A is in $\text{Labs}(\phi) \setminus \mathcal{N}(\phi) = \text{cand}(\phi)$, which means $A \subseteq \text{cand}(\phi)$, i.e. A is homogeneous.

Secondly, assume some $A \subseteq \text{Labs}(\phi)$ is homogeneous. Take an arbitrary $\ell \in \text{Labs}(\phi)$. The following must then be true:

$$\ell \in \mathcal{N}(\phi) \Rightarrow \ell \notin A \iff \mathcal{M}[A] \nvDash b_\ell \iff \mathcal{M}[A] \vDash \neg b_\ell$$

So we can conclude that $\mathcal{M}[A] \vDash \bigwedge_{\ell \in \mathcal{N}(\phi)} \neg b_\ell$, that is, $\mathcal{M}[A] \vDash \theta_H(\phi)$. ☐

Proposition 2. *For every* α-TLA$^+$ *expression* ϕ *and* $A \subseteq \text{Labs}(\phi)$*, it holds that* $\mathcal{M}[A] \vDash \theta_H(\phi) \land \theta^*_C(\phi)$ *if and only if* A *is a covering set for* ϕ*.*

Proof. Firstly, assume $\mathcal{M}[A] \vDash \theta_H(\phi) \land \theta^*_C(\phi)$. This obviously implies that $\mathcal{M}[A] \vDash \theta_H(\phi)$. By Proposition 1, we know A is homogeneous. We will prove that A is a covering set by contradiction:

Take an arbitrary branch $Br \in \text{Branches}(\phi)$ and $v' \in \mathit{Vars}'(\phi)$ and assume that $A \cap Br \cap \text{cand}(v', \phi)$ is empty. Because $\mathcal{M}[A] \vDash \theta^*_C(\phi)$ and $\theta^*_C(\phi) \Rightarrow \delta_{v'}(\phi)$, by definition, it must hold that $\mathcal{M}[A] \vDash \delta_{v'}(\phi)$. By Lemma 2, we know it suffices to consider only the labels from $A \cap \text{cand}(v', \phi)$, which we denote by $A|_{v'}$, for which $\mathcal{M}[A|_{v'}] \vDash \delta_{v'}(\phi)$. By Lemma 9, we can deduce that $\mathcal{M}[\mathcal{L} \setminus A|_{v'}] \vDash \neg\,\text{boolForm}(\phi)$. Since we assumed $Br \cap A|_{v'} = \emptyset$, it follows that $Br \subseteq \mathcal{L} \setminus A|_{v'}$. Because of this we can apply Lemma 10, as $\neg\,\text{boolForm}(\phi)$ in NNF contains only negated atoms, to conclude $\mathcal{M}[Br] \vDash \neg\,\text{boolForm}(\phi)$. However, as Br is a branch it must hold that $\mathcal{M}[Br] \vDash \text{boolForm}(\phi)$ as well, which is a contradiction.

Therefore, $Br \cap A|_v$ must be nonempty. As both Br and v' were arbitrary this implies that A is a covering set.

Secondly, consider the opposite direction, where $A \subseteq \mathcal{L}$ is a covering set. We must show that $\mathcal{M}[A] \vDash \theta^*_C(\phi)$, since covering sets are homogeneous, which implies $\mathcal{M}[A] \vDash \theta_H(\phi)$ by Proposition 1. It suffices to see that for every $v' \in \mathit{Vars}'(\phi)$ it holds that $\mathcal{M}[A] \vDash \delta_{v'}(\phi)$. We prove the following statement by induction on the structure of ϕ: For every variable $v' \in \mathit{Vars}'(\phi)$, equation (1) holds:

$$(\forall Br \in \text{ Branches}(\phi) \ . \ A \cap Br \cap \text{cand}(v', \phi) \neq \emptyset) \Rightarrow \mathcal{M}[A] \vDash \delta_{v'}(\phi) \qquad (1)$$

- $\phi = \ell :: \text{FALSE}$: Since $\text{Branches}(\phi) = \{\{\ell\}\}$ and $\ell \notin \text{cand}(\phi)$, no set can satisfy the precondition, so the implication vacuously holds.
- $\phi = \ell :: \star(v'_1, \ldots, v'_k)$: Same as above.
- $\phi = \ell :: w' \in \phi_1$: We know $\text{Branches}(\phi) = \{\{\ell\}\}$. Take an arbitrary $v' \in \mathit{Vars}'(\phi)$ and assume the precondition $\forall Br \in \text{ Branches}(\phi) \ . \ A \cap Br \cap \text{cand}(v', \phi) \neq \emptyset$. If $v' \neq w'$ then the precondition cannot hold, so (1) holds vacuously. Alternatively, if $v' = w'$, we deduce that A must contain $\{\ell\}$. Since $\delta_{w'}(\phi) = b_\ell$, clearly, $\mathcal{M}[A] \vDash b_\ell$.
- $\phi = \ell :: \bigwedge_{i=1}^{s} \phi_i$: Assume as the induction hypothesis, that (1) holds for every $\phi_k, k \in \{1, \ldots, s\}$. Take an arbitrary $v' \in \mathit{Vars}'(\phi)$ and assume the precondition that

  $$\forall Br \in \text{ Branches}(\phi) \ . \ Br \cap [A \cap \text{cand}(v', \phi)] \neq \emptyset$$

  By applying Lemma 8, with $J = A \cap \text{cand}(v', \phi)$, we can deduce that there is some $k \in \{1, \ldots, s\}$, for which it holds that

  $$\forall Br \in \text{ Branches}(\phi_k) \ . \ Br \cap [A \cap \text{cand}(v', \phi)] \neq \emptyset$$

  Since any label that is both in Br, a branch of ϕ_k, and $\text{cand}(v', \phi)$ is in $\text{cand}(v', \phi_k)$, we see that $B \cap A \cap \text{cand}(v', \phi_k)$ is also nonempty. By the induction hypothesis for ϕ_k, this tells us that $\mathcal{M}[A] \vDash \delta_{v'}(\phi_k)$. Since, by definition, $\delta_{v'}(\phi) = \bigvee_{i=1}^{s} \delta_{v'}(\phi_i)$, it must hold that $\mathcal{M}[A] \vDash \delta_{v'}(\phi)$.
- $\phi = \ell :: \bigvee_{i=1}^{s} \phi_i$: Assume as the induction hypothesis, that (1) holds for every $\phi_k, k \in \{1, \ldots, s\}$. Take an arbitrary $v' \in \mathit{Vars}'(\phi)$ and assume the precondition that

  $$\forall Br \in \text{ Branches}(\phi) \ . \ Br \cap [A \cap \text{cand}(v', \phi)] \neq \emptyset$$

  By applying Lemma 5, we see that $\text{Branches}(\phi) = \bigcup_{i=1}^{s} \text{Branches}(\phi_i)$. We can deduce

  $$\forall k \in \{1, \ldots, s\} \ . \ \forall Br \in \text{ Branches}(\phi_k) \ . \ Br \cap [A \cap \text{cand}(v', \phi)] \neq \emptyset$$

  By the same argument as in the conjunctive case, any label in $Br \cap \text{cand}(v', \phi)$, where $Br \in \text{Branches}(\phi_k)$, is also in $\text{cand}(v', \phi_k)$, so by using the induction hypothesis, we conclude $M[A] \vDash \delta_{v'}(\phi_k)$ for all $k \in \{1, \ldots, s\}$.
  This means $\mathcal{M}[A] \vDash \bigwedge_{i=1}^{s} \delta_{v'}(\phi_k)$ so as $\delta_{v'}(\phi) = \bigwedge_{i=1}^{s} \delta_{v'}(\phi_k)$ we see that $\mathcal{M}[A] \vDash \delta_{v'}(\phi)$.
- $\phi = \ell :: \exists x \in \phi_1 \colon \phi_2$: Assume as the induction hypothesis, that (1) holds for ϕ_2. Take an arbitrary $v' \in \mathit{Vars}'(\phi)$ and assume the precondition that

  $$\forall Br \in \text{ Branches}(\phi) \ . \ Br \cap [A \cap \text{cand}(v', \phi)] \neq \emptyset$$

By applying Lemma 6, we see that $\text{Branches}(\phi) = \text{Branches}(\phi_2)$, so it is clear that $\mathcal{M}[A] \vDash \delta_{v'}(\phi_2)$ by the induction hypothesis. Since $\delta_{v'}(\phi) = \delta_{v'}(\phi_2)$ we know that $\mathcal{M}[A] \vDash \delta_{v'}(\phi)$.

– $\phi = \ell$ :: IF ϕ_1 THEN ϕ_2 ELSE ϕ_3 : Assume as the induction hypothesis, that the statement holds for ϕ_2, ϕ_3. Take an arbitrary $v' \in \mathit{Vars}'(\phi)$ and assume the precondition that

$$\forall Br \in \text{Branches}(\phi) \ . \ Br \cap [A \cap \text{cand}(v', \phi)] \neq \emptyset$$

By applying Lemma 7, we see that $\text{Branches}(\phi) = \text{Branches}(\phi_2) \cup \text{Branches}(\phi_3)$. The rest of this proof is the same as for the disjunctive case, since $\delta_{v'}(\phi) = \delta_{v'}(\phi_2) \wedge \delta_{v'}(\phi_3)$ and $\text{boolForm}(\phi) = \text{boolForm}(\phi_2) \vee \text{boolForm}(\phi_3)$.

So we can conclude, that (1) always holds. Take an arbitrary $v' \in \mathit{Vars}'(\phi)$. Since A is a covering set, if also satisfies the precondition of (1). Therefore, we know that $\mathcal{M}[A] \vDash \delta_{v'}(\phi)$. As v' was arbitrary, it must be the case that $\mathcal{M}[A] \vDash \theta^*_C(\phi)$. □

Proposition 3. *For every α-TLA$^+$ expression ϕ and $A \subseteq \text{Labs}(\phi)$, it holds that $\mathcal{M}[A] \vDash \theta_H(\phi) \wedge \theta_C(\phi)$ if and only if A is a minimal covering set for ϕ.*

Proof. Firstly, assume $\mathcal{M}[A] \vDash \theta_H(\phi) \wedge \theta_C(\phi)$. We already know, from Proposition 2, that such an A is a covering set, since $\mathcal{M}[A] \vDash \theta^*_C(\phi)$ is implied.

Take an arbitrary $Br \in \text{Branches}(\phi)$ and $v' \in \mathit{Vars}'(\phi)$. We must show that $|A \cap Br \cap \text{cand}(v', \phi)| = 1$. We know the set is nonempty, so consider an arbitrary pair $i, j \in A \cap Br \cap \text{cand}(v', \phi)$. Clearly, $\{i, j\} \subseteq Br$ and $\{i, j\} \subseteq \text{cand}(v', \phi)$ so we know $(i, j), (j, i) \in \text{Colloc}_{v'}(\phi)$. We demonstrate that $i = j$ by contradiction.

Assume that $i \neq j$ and w.l.o.g. $i < j$. Since $\mathcal{M}[A] \vDash \theta^{\exists!}(\phi)$ and, by assumption, $i < j$, we must have a term $\neg(b_i \wedge b_j)$, and can conclude $\mathcal{M}[A] \vDash \neg b_i \vee \neg b_j$. However, this is only true if $i \notin A \vee j \notin A$. As we have selected i, j such that $i, j \in A$ we have a contradiction. It then follows that $i = j$ and the intersection is a singleton, as required.

Secondly, assume that a set A is a minimal covering set. In particular, it is also a covering set and thus $\mathcal{M}[A] \vDash \theta_H(\phi) \wedge \theta^*_C(\phi)$, by Proposition 2. To show that $\mathcal{M}[A] \vDash \theta^{\exists!}(\phi)$, take an arbitrary $v' \in \mathit{Vars}'(\phi)$ and $i < j$ for which $(i, j) \in \text{Colloc}_{v'}(\phi)$. We need to see that $\mathcal{M}[A] \vDash \neg(b_i \wedge b_j)$. By definition, there exists a $Br \in \text{Branches}(\phi)$, for which $\{i, j\} \subseteq Br$, as $\text{Colloc}_{v'}(\phi) \subseteq \text{Colloc}(\phi)$. Since A is a minimal covering set, we know $A \cap Br \cap \text{cand}(v', \phi)$ is a singleton. Both i and j belong to $B \cap \text{cand}(v', \phi)$ and $i < j$ implies that they are distinct, so one of them must not belong to A. This means $\mathcal{M}[A] \vDash \neg b_i \vee \neg b_j$. As this holds for an arbitrary selection of v' and (i, j), clearly $\mathcal{M}[A] \vDash \theta^{\exists!}(\phi)$, which we needed to show. □

Proposition 4. *For every α-TLA$^+$ expression ϕ and $A \subseteq \text{Labs}(\phi)$, there is a function $r \colon \mathbb{N} \to \mathbb{N}$, for which $(\mathcal{M}[A], r) \vDash \theta_H(\phi) \wedge \theta_A(\phi)$ if and only if A is acyclic.*

Proof. Firstly, assume that there exists an $r \colon \mathbb{N} \to \mathbb{N}$, for which $(\mathcal{M}[A], r) \vDash \theta_H(\phi) \wedge \theta_A(\phi)$. This obviously implies that $\mathcal{M}[A] \vDash \theta_H(\phi)$. By Proposition 1, we know A is homogeneous. We define $\prec_A$ using r:

$$\ell_1 \prec_A \ell_2 \iff r(\ell_1) < r(\ell_2)$$

Clearly, $<$ is a strict total order on $\mathbb{N}$. We have ensured that

$$\bigwedge_{\substack{\ell_1,\ell_j \in \text{Labs}(\phi) \\ \ell_i < \ell_j}} r(\ell_i) \neq r(\ell_j)$$

so r restricted to $\text{Labs}(\phi)$ is injective. As $A \subseteq \text{Labs}(\phi)$ we know that r restricted to A is injective as well. We can then use Lemma 11, for the function $f = r|_A : A \to \mathbb{N}$, to conclude that such a $\prec_A$ is a strict total order on A. Now take an arbitrary branch $Br \in \text{Branches}(\phi)$, two labels $\ell_1, \ell_2 \in A \cap Br$ and a variable $v' \in \mathit{Vars}'(\phi)$. If the relation $\ell_1 \lhd_{v'} \ell_2$ does not hold, the implication $\ell_1 \lhd_{v'} \ell_2 \Rightarrow \ell_1 \prec_A \ell_2$ is vacuously correct. If it does, since ℓ_1, ℓ_2 belong to $A \cap Br$, we know that $(\ell_1, \ell_2) \in \text{Colloc}_\lhd(\phi)$. As $(\mathcal{M}[A], r) \vDash \theta_A(\phi)$ it is also the case that $(\mathcal{M}[A], r) \vDash \theta^*_A(\phi)$. We know that $\mathcal{M}[A] \vDash b_{\ell_1} \wedge b_{\ell_2}$ so it must be the case that $r(\ell_1) < r(\ell_2)$. But then, by definition, $\ell_1 \prec_A \ell_2$. Because Br, ℓ_1, ℓ_2 and v' were arbitrary, we can conclude that A is acyclic.

Secondly, assume A is acyclic. We must show that $\mathcal{M}[A] \vDash \theta^*_A(\phi)$, since acyclic sets are homogeneous, which implies $\mathcal{M}[A] \vDash \theta_H(\phi)$ by Proposition 1. We can take the strict total order $\prec_A$ and extend it to a strict total order $\prec$ on $\text{Labs}(\phi)$. Because of this, there exists an ordering function $\text{ord} \colon \text{Labs}(\phi) \to \{1, \ldots, |\text{Labs}(\phi)|\}$ with the property

$$\ell_1 \prec \ell_2 \iff \text{ord}(\ell_1) < \text{ord}(\ell_2)$$

we can define $r \colon \mathbb{N} \to \mathbb{N}$ as

$$r(n) = \begin{cases} \text{ord}(n) & ; n \in \text{Labs}(\phi) \\ 1 & ; \text{otherwise} \end{cases}$$

Let us first see that $(\mathcal{M}[A], r) \vDash \theta^*_A(\phi)$. Take an arbitrary pair $(i, j) \in \text{Colloc}_\lhd(\phi)$. We need to show that $(\mathcal{M}[A], r) \vDash b_i \wedge b_j \Rightarrow R(i) < R(j)$. If $i \notin A$ or $j \notin A$ then $b_i \wedge b_j$ evaluates to false and the implication in satisfied. If both i and j belong to A, then we take an arbitrary $Br \in \text{Branches}(\phi)$ containing both of them (it exists, since $(i, j) \in \text{Colloc}(\phi)$ as $\text{Colloc}_\lhd(\phi) \subseteq \text{Colloc}(\phi)$) and the variable $v' \in V$ for which $i \lhd_{v'} j$. As A is acyclic, we can instantiate the acyclicity criterion for our choice of Br, i, j and v' and conclude $i \prec_A j$. Because $\prec$ extends $\prec_A$ it must be the case that $\text{ord}(i) < \text{ord}(j)$ and, because $r|_{\text{Labs}(\phi)} = \text{ord}$, also $r(i) < r(j)$. So $(\mathcal{M}[A], r)$ models $\theta^*_A(\phi)$. We conclude the proof by showing that this r also models the formula

$$\bigwedge_{\substack{\ell_1,\ell_j \in \text{Labs}(\phi) \\ \ell_i < \ell_j}} R(\ell_i) \neq R(\ell_j)$$

If $\ell_1, \ell_2 \in \text{Labs}(\phi)$ then $r(\ell_1) = \text{ord}(\ell_1)$ and $r(\ell_2) = \text{ord}(\ell_2)$. It then follows, as ord is bijective, that either $r(\ell_1) < r(\ell_2)$ or vice-versa. In any case, $r(\ell_1) \neq r(\ell_2)$. Altogether, this implies $(\mathcal{M}[A], r) \vDash \theta_A(\phi)$. ☐

B.3 Proofs of Section 6

Proposition 5. *Let ϕ be an α-TLA$^+$ expression and $M = (\mathcal{I}, \xi, s, s')$ a model of the TLA$^+$ formula $\gamma(\phi)$. There exists a branch Br of ϕ such that M is also a model of $\gamma(\text{slice}(\phi, Br))$.*

Proof. We prove this proposition by induction on depth of an α-TLA$^+$ formula ϕ.

Base case depth $(\phi) = 0$. We have that boolForm $(\phi) = b_{\ell_1}$. Therefore there exists exactly one branch $Br_0 = \{\ell_1\}$ on ϕ. It implies $\text{slice}(\phi, Br_0) = \phi$. Because M is a model of $\gamma(\phi)$, we know that M is also a model of $\gamma(\text{slice}(\phi, Br_0))$.

Assume that the theorem holds for depth $(\phi) \leq k$. We will show it for the case *depth* $(\phi) = k + 1$. There are four cases:

a) Case $\phi = \ell :: \phi_1 \vee \phi_2$.
We know that $M \vDash \gamma(\phi_1) \vee \gamma(\phi_2)$. Without lost of generality, assume that $M \vDash \gamma(\phi_1)$. Applying the induction hypothesis, there exists a branch Br_1 of ϕ_1 such that $M \vDash \gamma(\text{slice}(\phi_1, Br_1))$. By Lemma 5, we know that Br_1 is also a branch of ϕ. Because $\gamma(\text{slice}(\phi, Br_1)) = \gamma(\text{slice}(\phi_1, Br_1)) \vee \gamma(\text{slice}(\phi_2, Br_1))$, we have that $M \vDash \gamma(\text{slice}(\phi, Br_1))$.

b) Case $\phi = \ell :: \phi_1 \wedge \phi_2$.
We have boolForm (ϕ) = boolForm $(\phi_1) \wedge$ boolForm (ϕ_2) and $M \vDash \gamma(\phi_i)$ for every $i \in \{1, 2\}$. Applying the induction hypothesis to every subformula ϕ_i, we know that there exists a branch Br_i for every ϕ_i such that $M \vDash \gamma(\text{slice}(\phi_i, Br_i))$. Let Br is the union of Br_1 and Br_2. By Lemma 4, we have that Br is a branch of ϕ. By Lemma 3, we have

$$\begin{aligned}\text{slice}(\phi, Br) &= \text{slice}(\phi_1, Br) \wedge \text{slice}(\phi_2, Br)\\ &= \text{slice}(\phi_1, Br_1 \cup Br_2) \wedge \text{slice}(\phi_2, Br_1 \cup Br_2)\\ &= \text{slice}(\phi_1, Br_1) \wedge \text{slice}(\phi_2, Br_2)\end{aligned}$$

Since $M \vDash \gamma(\text{slice}(\phi_i, Br_i))$ for every $i \in \{1, 2\}$, we have that M is a model of $\gamma(\text{slice}(\phi, Br))$.

c) Case $\phi = \ell :: \exists x \in S \,.\, \phi_1$.
Notice that if $\gamma(S)$ is the empty set, there is no model for $\gamma(\phi)$. Since M is a model of $\gamma(\phi)$, we know that $\gamma(S)$ is not the empty set and therefore, there exists x_0 in S such that $M \vDash \gamma(\phi_1)[x \leftarrow x_0]$. Because of the induction hypothesis, we know that there exists a branch Br_1 such that $M \vDash \gamma(\text{slice}(\phi_1, Br_1))[x \leftarrow x_0]$. Moreover, by Lemma 6 we have that Br_1 is also a branch of ϕ and therefore, $\text{slice}(\phi, Br_1) = \exists x \in S \,.\, \text{slice}(\phi_1, Br_1)$. Because $x_0 \in \gamma(S)$, we have that $M \vDash \gamma(\text{slice}(\phi, Br_1))$.

d) Case $\phi = \ell :: \text{IF } \phi_1 \text{ THEN } \phi_2 \text{ ELSE } \phi_3$.
The are two subcases: $M \vDash \gamma(\phi_1 \wedge \phi_2)$, or $M \vDash \gamma(\neg\phi_1 \wedge \phi_3)$. In both cases, the arguments are similar to the conjunction case.

In conclusion, we have that the theorem is true for all depth (ϕ), or for all logical formula ϕ. ∎

Proposition 6. *Let ϕ be an α-TLA$^+$ expression and $M = (\mathcal{I}, \xi, s, s')$ a model of the* TLA$^+$ *formula $\gamma\,(\text{slice}(\phi, Br))$. Then, M is also a model of $\gamma\,(\phi)$.*

Proof. We prove this proposition by induction on depth of an α-TLA$^+$ formula ϕ.

Base case depth $(\phi) = 0$. We have that boolForm (ϕ) contains only b_{ℓ_1}. Therefore there exists exactly one branch $Br_0 = \{\ell_1\}$ on ϕ. It implies $\text{slice}(\phi, Br_0) = \phi$. It means M is a model of $\gamma\,(\phi)$.

Assume that the theorem holds for depth $(\phi) \leq k$. We will show it for the case depth $(\phi) = k + 1$. There are four cases:

a) Case $\phi = \ell :: \phi_1 \vee \phi_2$. We know $\gamma(\phi) = \gamma(\phi_1) \vee \gamma(\phi_2)$. By definition, $\gamma(\text{slice}(\phi, Br)) = \gamma(\text{slice}(\phi_1, Br)) \vee \gamma(\text{slice}(\phi_2, Br))$ Applying Lemma 5 we know that Br is a branch of either ϕ_1 or of ϕ_2. Without loss of generality, it is a branch of ϕ_1. Then, it follows that $\text{slice}(\phi_1, Br) = \phi_1$ and $\gamma(\text{slice}(\phi_2, Br)) = \text{FALSE}$ by Lemma 3, as $\text{Labs}(\phi_2) \cap Br = \emptyset$. Therefore $\gamma(\text{slice}(\phi, Br))$ is equivalent to $\gamma(\phi_1)$. As M is a model of $\gamma(\text{slice}(\phi, Br))$ it is also a model of $\gamma(\phi_1)$ and consequently a model of $\gamma(\phi)$

b) Case $\phi = \ell :: \phi_1 \wedge \phi_2$.
We know boolForm (ϕ) = boolForm $(\phi_1) \wedge$ boolForm (ϕ_2). Since Br is a branch of ϕ, by Lemma 4 we have that $Br_i = \{b_\ell \mid b_\ell \in Br \wedge \ell \in \text{Labs}\,(\phi_i)\}$ is a branch of ϕ_i for every $i \in \{1, 2\}$. Moreover, by Lemma 4 we know that $Br = Br_1 \cup Br_2$. By Lemma 3, we have

$$\begin{aligned}\text{slice}(\phi, Br) &= \text{slice}(\phi_1, Br) \wedge \text{slice}(\phi_2, Br) \\ &= \text{slice}(\phi_1, Br_1) \wedge \text{slice}(\phi_2, Br_2)\end{aligned}$$

Since $M_1 \vDash \gamma\,(\text{slice}(\phi, Br))$, we have that $M_1 \vDash \gamma\,(\text{slice}(\phi_i, B_i))$ for every $i \in \{1, 2\}$. According to the induction hypothesis, we know that M_1 is a model of both $\gamma\,(\phi_1)$ and $\gamma\,(\phi_2)$. It implies M_1 is a model of $\gamma\,(\phi)$.

c) Case $\phi = \ell :: \exists x \in S\,.\,\phi_1$.
We have that $M_1 \vDash \gamma\,(\text{slice}(\phi, Br))$ or $M_1 \vDash \gamma\,(\exists x \in S\,.\,\text{slice}(\phi_1, Br))$. If $\gamma\,(S)$ is the empty set, there is no model for $\gamma\,(\exists x \in S\,.\,\text{slice}(\phi_1, Br))$. Therefore, $\gamma\,(S)$ is is not the empty set. Let x_0 be an arbitrary element in $\gamma\,(S)$ such that $M_1 \vDash \gamma\,(\text{slice}(\phi_1, Br))\,[x \leftarrow x_0]$. Notice that by Lemma 6 we know Br is also a branch of ϕ_1. Applying the induction hypothesis, we have that M_1 is a model of $\gamma\,(\phi_1)\,[x \leftarrow x_0]$. It implies that M_1 is a model of $\gamma\,(\phi_2\,[x \leftarrow x_0])$. Because x_0 is a element of $\gamma\,(S)$, we have that M_1 is a model of $\gamma\,(\phi)$.

d) Case $\phi = \ell :: \text{IF } \phi_1 \text{ THEN } \phi_2 \text{ ELSE } \phi_3$.
The are two subcases: M_1 is a model of $\gamma\,(\phi_1 \wedge \text{slice}(\phi_2, Br))$, or M_1 is a model of $\gamma\,(\neg\phi_1 \wedge \text{slice}(\phi_3, Br))$. By Lemma 7 we have that Br is a branch of ϕ_2 in the first case and that Br is a branch of ϕ_3 in the second case. From the induction hypothesis, it is easy to show that M_1 is also a model of $\gamma\,(\phi)$.

In conclusion, we have that the theorem is true for all depth (ϕ), or for all logical formula ϕ. □

Proposition 7. *Let ϕ be an α-TLA$^+$ expression. For every $S, T \subseteq \text{Labs}(\phi)$, every model M of the* TLA$^+$ *formula $\gamma\,(\text{slice}(\phi, S))$, is also a model of $\gamma\,(\text{slice}(\phi, S \cup T))$.*

Proof. If $\gamma(S)$ is the empty set, then $\gamma(\text{slice}(\phi, S)) = \text{FALSE}$. Therefore, there is no model of $\gamma(\text{slice}(\phi, S))$, and this proposition holds.

If $T = \emptyset$, we have that $\text{slice}(\phi, S \cup T) = \text{slice}(\phi, S)$. Therefore, M is also a model of $\gamma(\text{slice}(\phi, S \cup T))$.

If both $S \neq \emptyset$ and $T \neq \emptyset$, we prove this proposition by induction on depth of an α-TLA$^+$ formula ϕ.

Base case $\text{depth}(\phi) = 0$. We have that $\text{boolForm}(\phi)$ contains only b_{ℓ_1}. Therefore, we know that $S = T = S \cup T = \{b_{\ell_1}\}$. It implies that M is a model of $\gamma(\text{slice}(\phi, S \cup T))$.

Assume that the theorem holds for $\text{depth}(\phi) \leq k$. We will show it for the case $\text{depth}(\phi) = k + 1$. There are four cases:

a) Case $\phi = \ell :: \phi_1 \vee \phi_2$.
Because $M \models \gamma(\text{slice}(\phi, S))$ and $\gamma(\text{slice}(\phi, S)) \Leftrightarrow \gamma(\text{slice}(\phi_1, S)) \vee \gamma(\text{slice}(\phi_2, S))$, we know that M is a model of either $\gamma(\text{slice}(\phi_1, S))$ or $\gamma(\text{slice}(\phi_2, S))$. If M is a model of $\gamma(\text{slice}(\phi_1, S))$, by the induction hypothesis we have M is a model of $\gamma(\text{slice}(\phi_1, S \cup T))$. If M is a model of$\gamma(\text{slice}(\phi_2, S))$, by the induction hypothesis we have M is a model of $\gamma(\text{slice}(\phi_2, S \cup T))$. Therefore, we know that M is a model of $\gamma(\text{slice}(\phi, S \cup T))$ since

$$\gamma(\text{slice}(\phi, S \cup T)) \Leftrightarrow \gamma(\text{slice}(\phi_1, S \cup T)) \vee \gamma(\text{slice}(\phi_2, S \cup T))$$

b) Case $\phi = \ell :: \phi_1 \wedge \phi_2$.
Because $M \models \gamma(\text{slice}(\phi, S))$ and $\gamma(\text{slice}(\phi, S)) \Leftrightarrow \gamma(\text{slice}(\phi_1, S)) \wedge \gamma(\text{slice}(\phi_2, S))$, we know that M is a model of both $\gamma(\text{slice}(\phi_1, S))$ and $\gamma(\text{slice}(\phi_2, S))$. By the induction hypothesis, we have that M is a model of both $\gamma(\text{slice}(\phi_1, S \cup T))$ and $\gamma(\text{slice}(\phi_2, S \cup T))$. Therefore, we know that M is a model of $\gamma(\text{slice}(\phi, S \cup T))$ since

$$\gamma(\text{slice}(\phi, S \cup T)) \Leftrightarrow \gamma(\text{slice}(\phi_1, S \cup T)) \wedge \gamma(\text{slice}(\phi_2, S \cup T))$$

c) Case $\phi = \ell :: \exists x \in \phi_1 \,.\, \phi_2$.
Because $\text{slice}(\phi, S) = \exists x \in \phi_1 \,.\, \text{slice}(\phi_2, S)$, we know that if $\gamma(\phi_1)$ is the empty set, there is no model of $\gamma(\text{slice}(\phi, S))$. Let x_0 be an element in $\gamma(\phi_1)$ such that M is a model of $\gamma(\text{slice}(\phi, S))\,[x \leftarrow x_0]$. Applying the induction hypothesis, we have that $M \models \gamma(\text{slice}(\phi, S \cup T))\,[x \leftarrow x_0]$. Because $x_0 \in \gamma(\phi_1)$, we have $M \models \gamma(\exists x \in \phi_1 \,.\, \text{slice}(\phi, S \cup T))$.

d) Case $\phi = \ell ::\ \text{IF}\ \phi_1\ \text{THEN}\ \phi_2\ \text{ELSE}\ \phi_3$.
There are two cases here:
- If $M \models \gamma(\phi_1) \wedge \gamma(\text{slice}(\phi_2, S))$, then by the induction hypothesis, we have that $M \models \gamma(\phi_1) \wedge \gamma(\text{slice}(\phi_2, S \cup T))$. Therefore, we have $M \models \text{slice}(\phi, S \cup T)$.
- If $M \models \gamma(\neg\phi_1) \wedge \gamma(\text{slice}(\phi_3, S))$, then by the induction hypothesis, we have $M \models \gamma(\neg\phi_1) \wedge \gamma(\text{slice}(\phi_3, S \cup T))$. Therefore, we have $M \models \text{slice}(\phi, S \cup T)$.

In conclusion, we have that the theorem is true for all $\text{depth}(\phi)$, or for all logical formula ϕ. □

Proposition 8. *Let ϕ be an α-TLA$^+$ expression. For any selection $Br_1, \ldots, Br_k$ from the branches of ϕ, the following holds: If there exists a model M of the formula $\gamma(\text{slice}(\phi, Br_1 \cup \cdots \cup Br_k))$, then M must be a model of $\gamma(\text{slice}(\phi, Br))$, for some branch $Br \in \text{Branches}(\phi)$. Additionally, if there is an assignment strategy A for ϕ, such that $Br_1, \ldots, Br_k$ all belong to the same equivalence class $[B]_A$, then M must be a model of $\gamma(\text{slice}(\phi, Br))$, for some branch $Br \in [B]_A$.*

Proof. Denote by S the union $Br_1 \cup \cdots \cup Br_k$. We prove the lemma by induction on the structure of ϕ:

- $\phi = \ell :: \text{FALSE}$: Since, for any $T \subseteq \text{Labs}(\phi)$ the formula $\gamma(\text{slice}(\phi, T))$ is equivalent to FALSE, it does not have a model, so the implication vacuously holds.
- $\phi = \ell :: \star\,(v'_1, \ldots, v'_k)$: Assume there exists a model M of $\gamma(\text{slice}(\phi, S))$. This means that ℓ must belong to S, otherwise $\text{slice}(\phi, S)$ is $\ell :: \text{FALSE}$ and γ applied to $\ell :: \text{FALSE}$ is FALSE, which does not have a model. As ϕ has exactly one branch, the rest of the lemma follows trivially, since $S \supseteq Br_1$. Consequently, $\text{slice}(\phi, Br_1)$, ϕ and $\text{slice}(\phi, S)$ are all the same expression, by Lemma 3. As M is a model of $\gamma(\text{slice}(\phi, S))$ it is also a model of $\text{slice}(\phi, Br_1)$. It is clear that if all chosen branches belong to $[B]_A$, then, in particular, $Br_1 \in [B]_A$.
- $\phi = \ell :: w' \in \phi_1$: Same as the previous case.
- $\phi = \ell :: \bigwedge_{i=1}^{s} \phi_i$: Assume, as the induction hypothesis, that the lemma holds for all ϕ_i. By definition, $\text{slice}(\phi, S) = \ell :: \bigwedge_{i=1}^{s} \text{slice}(\phi_i, S)$. Assume, that M is a model of $\gamma(\text{slice}(\phi, S))$. Then, $\gamma(\text{slice}(\phi, S)) = \bigwedge_{i=1}^{s} \gamma(\text{slice}(\phi_i, S))$ and M is a model of $\gamma(\text{slice}(\phi_i, S))$ for every i. By Lemma 4, we know that for each $i = 1, \ldots, k$ there exist branches $Br_i^1, \ldots, Br_i^s$ of $\phi_1, \ldots, \phi_s$, such that

  $$Br_i = \bigcup_{j=1}^{s} Br_i^j$$

  Take an arbitrary $i \in \{1, \ldots, s\}$. By Lemma 3, we know that

  $$\text{slice}(\phi_i, S) = \text{slice}(\phi_i, S \cap \text{Labs}(\phi_i))$$

  We can see the following:

  $$\begin{aligned} S \cap \text{Labs}(\phi_i) &= \left(\bigcup_{t=1}^{k} Br_t\right) \cap \text{Labs}(\phi_i) \\ &= \left(\bigcup_{t=1}^{k} \bigcup_{j=1}^{s} Br_t^j\right) \cap \text{Labs}(\phi_i) \\ &= \bigcup_{t=1}^{k} \bigcup_{j=1}^{s} \left(Br_t^j \cap \text{Labs}(\phi_i)\right) \end{aligned}$$

As each Br_t^j is a branch of ϕ_j, all of the intersections are either Br_t^j, if $i = j$, or empty. Consequently:

$$S \cap \text{Labs}(\phi_i) = \bigcup_{t=1}^{k} Br_t^i$$

By design, Br_t^i is a branch of ϕ_i, for each t. This means we can apply our induction hypothesis for ϕ_i to deduce that there must exist a $B^i \in \text{Branches}(\phi_i)$, for which M is a model of $\gamma(\text{slice}(\phi_i, B^i))$. As i was arbitrary, this holds for every selection of i. We thus obtain a collection of branches, $B^1, \ldots, B^s$, where it holds that M is a model of $\gamma(\text{slice}(\phi_i, B^i))$ for every $i \in \{1, \ldots, s\}$. Using $B_0 = \bigcup_{i=1}^{s} B^i$ and Proposition 7, we deduce that M is a model of $\gamma(\text{slice}(\phi_i, B_0))$, for each $i \in \{1, \ldots, s\}$. By Lemma 4, B_0 is a branch of ϕ. So it follows that M is a model of $\gamma(\text{slice}(\phi, B_0))$.
Assume additionally that $Br_1, \ldots, Br_k \in [B]$ for some assignment strategy A and some equivalence class $[B]$ of $\sim_A$.
By definition, $Br_i \cap A = Br_1 \cap A$ for all $i \in \{1, \ldots, k\}$. It follows, that $(Br_i \cap \text{Labs}(\phi_j)) \cap A = (Br_1 \cap \text{Labs}(\phi_j)) \cap A$ for all $i \in \{1, \ldots, k\}$ and all $j \in \{1, \ldots s\}$. This means that the sets $Br_1^i, \ldots, Br_k^i$ are equivalent for all $i \in \{1, \ldots, s\}$, since $Br_j \cap \text{Labs}(\phi_i) = Br_j^i$. By the induction hypothesis, this implies that B^i is equivalent to Br_1^i, for all $i \in \{1, \ldots, s\}$. Altogether:

$$\begin{aligned} B_0 \cap A = &\bigcup_{i=1}^{s} (B^i \cap A) \\ = &\left(\bigcup_{i=1}^{s} Br_1^i \cap A \right) \\ = &\left(\bigcup_{i=1}^{s} Br_1^i \right) \cap A \\ = &Br_1 \cap A \end{aligned}$$

Thus we see that B_0 is equivalent to Br_1 and, by transitivity, to all of the branches $Br_1, \ldots, Br_k$.

– $\phi = \ell :: \bigvee_{i=1}^{s} \phi_i$: Assume, as the induction hypothesis, that the lemma holds for all ϕ_i. By definition, $\text{slice}(\phi, S) = \ell :: \bigvee_{i=1}^{s} \text{slice}(\phi_i, S)$. Then, $\gamma(\text{slice}(\phi, S)) = \bigvee_{i=1}^{s} \gamma(\text{slice}(\phi_i, S))$. If we assume that M is a model of $\gamma(\text{slice}(\phi, S))$, there must exist an $i \in \{1, \ldots, k\}$, for which M is a model of $\gamma(\text{slice}(\phi_i, S))$ By Lemma 3, we know that

$$\text{slice}(\phi_i, S) = \text{slice}(\phi_i, S \cap \text{Labs}(\phi_i))$$

Additionally, Lemma 5 guarantees that for each $j = 1, \ldots, k$ the set $Br_j \cap \text{Labs}(\phi_i)$ is either Br_j or empty. Because $\gamma(\text{slice}(\phi_i, S \cap \text{Labs}(\phi_i)))$ has a model, the set $S \cap \text{Labs}(\phi_i)$ is not empty. It is therefore a union of branches $Br'_1, \ldots, Br'_l$ from $\text{Branches}(\phi_i)$.

By the induction hypothesis for ϕ_i, we know that there exists a $B^i \in$ Branches(ϕ_i), for which M is a model of $\gamma(\text{slice}(\phi_i, B^i))$. By Lemma 5, B^i is also a branch for ϕ.
Assume additionally that $Br_1, \ldots, Br_k \in [B]_A$ for some assignment strategy A and some equivalence class $[B]_A$ of $\sim_A$. Trivially, all branches $Br'_1, \ldots, Br'_l$ are equivalent as well. Therefore, $B^i \sim_A Br'_1$. As Br'_1 is equal to one of the branches $Br_1, \ldots, Br_k$, which are all equivalent, we see that B^i is equivalent to Br_1 and, by transitivity, to all of the branches $Br_1, \ldots, Br_k$.

– $\phi = \ell :: \exists x \in \phi_1 \colon \phi_2$: Assume, as the induction hypothesis, that the lemma holds for ϕ_2. By definition, $\text{slice}(\phi, S) = \ell :: \exists x \in \phi_1 \colon \text{slice}(\phi_2, S)$. Assume, that M is a model of $\gamma(\text{slice}(\phi, S))$. Then,

$$\gamma(\text{slice}(\phi, S)) = \ell :: \exists x \in \gamma(\phi_1) \colon \gamma(\text{slice}(\phi_2, S))$$

It follows that $\gamma(\phi_1)$ is contains some x_0, for which M is a model of the formula $\gamma(\text{slice}(\phi_2, S))[x \leftarrow x_0]$. By Lemma 6, we know that branches of ϕ are exactly branches of ϕ_2, so each Br_i is a branch of ϕ_2. By the induction hypothesis, this means that there exists a $B \in$ Branches(ϕ_2), for which M is a model of $\gamma(\text{slice}(\phi_2, B))[x \leftarrow x_0]$. But this means that M is also a model of $\exists x \in \gamma(\phi_1) \colon \gamma(\text{slice}(\phi_2, B))$, since x_0 was chosen from $\gamma(\phi_1)$. Therefore it follows that M is a model for $\gamma(\text{slice}(\phi, B))$. Note that B is a branch of ϕ by Lemma 6.
Assume additionally that $Br_1, \ldots, Br_k \in [B]_A$ for some assignment strategy A and some equivalence class $[B]_A$ of $\sim_A$. As all branches $Br_1, \ldots, Br_l$ are branches of ϕ_2, the induction hypothesis guarantees that B is equivalent to Br_1 and, by transitivity, to all of the branches $Br_1, \ldots, Br_k$.

– $\phi = \ell :: \text{IF } \phi_1 \text{ THEN } \phi_2 \text{ ELSE } \phi_3$: Assume, as the induction hypothesis, that the lemma holds for ϕ_2 and ϕ_3. By definition,

$$\text{slice}(\phi, S) = \ell :: \text{IF } \phi_1 \text{ THEN } \text{slice}(\phi_2, S) \text{ ELSE } \text{slice}(\phi_3, S)$$

Assume, that M is a model of $\gamma(\text{slice}(\phi, S))$. Then,

$$\gamma(\text{slice}(\phi, S)) = \text{IF } \gamma(\phi_1) \text{ THEN } \gamma(\text{slice}(\phi_2, S)) \text{ ELSE } \gamma(\text{slice}(\phi_3, S))$$

By Lemma 3, we know that

$$\gamma(\text{slice}(\phi, S)) = \text{IF } \gamma(\phi_1) \text{ THEN } \gamma(\text{slice}(\phi_2, S \cap \text{Labs}(\phi_2))) \text{ ELSE } \gamma(\text{slice}(\phi_3, S \cap \text{Labs}(\phi_3)))$$

By Lemma 7, branches of ϕ are either branches of ϕ_2 or of ϕ_3. We have two options, either $\gamma(\phi_1)$ is true under M, or it isn't. If $\gamma(\phi_1)$ is true under M, then, as M is a model of $\gamma(\text{slice}(\phi, S))$, we can conclude that M is a model of $\gamma(\text{slice}(\phi_2, S \cap \text{Labs}(\phi_2)))$ and $S \cap \text{Labs}(\phi_2)$ is not empty. It is therefore a union of branches $Br'_1, \ldots, Br'_l$ from Branches(ϕ_2). By the induction hypothesis for ϕ_2, we know that there exists a $B^2 \in$ Branches(ϕ_2), for which M is a model of $\gamma(\text{slice}(\phi_2, B^2))$. By Lemma 7, B^2 is also a branch for ϕ.

Assume additionally that $Br_1, \dots, Br_k \in [B]_A$ for some assignment strategy A and some equivalence class $[B]_A$ of $\sim_A$. Trivially, all branches $Br'_1, \dots, Br'_l$ are equivalent as well. Therefore, $B^2 \sim_A Br'_1$. As Br'_1 is equal to one of the branches $Br_1, \dots, Br_k$, which are all equivalent, we see that B^2 is equivalent to Br_1 and, by transitivity, to all of the branches $Br_1, \dots, Br_k$.
The case where $\gamma(\phi_1)$ is false under M is proven analogously.

We conclude, that the lemma holds for all ϕ. □

Corollary 1. *Let ϕ be an α-TLA$^+$ expression and A an assignment strategy for ϕ. For every equivalence class $[Br]_A$ of $\sim_A$, the following holds: Using the set $X = \bigcup_{Y \in [Br]_A} Y$, if there exists a model M of $\gamma(\text{slice}(\phi, X))$, then M must be a model of $\gamma(\text{slice}(\phi, B))$, for some branch $B \in [Br]_A$.*

B.4 Proofs of Theorems

Theorem 1. *For every α-TLA$^+$ formula ϕ and $A \subseteq \text{Labs}(\phi)$, it holds that $\mathcal{M}[A] \vDash \theta(\phi)$ if and only if A is an assignment strategy for ϕ.*

Proof. Let ϕ be an α-TLA$^+$ formula and $A \subseteq \text{Labs}(\phi)$. By definition, $\theta(\phi) = \theta_H(\phi) \wedge \theta_C(\phi) \wedge \theta_A(\phi)$.

Firstly, assume $\mathcal{M}[A] \vDash \theta(\phi)$. As $\theta(\phi)$ implies both $\theta_H(\phi) \wedge \theta_C(\phi)$ and $\theta_H(\phi) \wedge \theta_A(\phi)$, we know A is a minimal covering and acyclic, by propositions 3 and 4 respectively. By definition, this means A is an assignment strategy.

Secondly, assume A is an assignment strategy. In particular, A a minimal covering, so $\mathcal{M}[A] \vDash \theta_H(\phi) \wedge \theta_C(\phi)$. Similarly, as A is acyclic, we know that $\mathcal{M}[A] \vDash \theta_H(\phi) \wedge \theta_A(\phi)$. It therefore follows that $\mathcal{M}[A] \vDash \theta_H(\phi) \wedge \theta_C(\phi) \wedge \theta_A(\phi)$, that is, $\mathcal{M}[A] \vDash \theta(\phi)$. □

Theorem 2. *Let ϕ be an α-TLA$^+$ expression and A an assignment strategy for ϕ. There is a model M of the TLA$^+$ formula $\gamma(\phi)$ if and only if there exists a $Br \in \text{Branches}(\phi)$, such that M is a model of $\gamma(\psi)$, where ψ is the symbolic transition generated by Br and A.*

Proof. First, assume that there exists a model M of $\gamma(\phi)$. By Proposition 5, we know that there exists a branch Br, for which M is a model of $\gamma(\text{slice}(\phi, Br))$. Then, denote by Y the set $\bigcup_{Z \in [Br]_A} Z$. Obviously, $Br \in [Br]_A$, so $Br \cup Y = Y$. It follows, by Proposition 7, that M is a model of $\gamma(\text{slice}(\phi, Y))$. But $\text{slice}(\phi, Y)$ is the symbolic transition generated by Br and A, by definition, so the implication holds.

Next, assume that there exists a model M of $\gamma(\psi)$, for a symbolic transition ψ. There exists an equivalence class $[Br]_A$, such that ψ has the shape $\text{slice}(\phi, Y)$ for $Y = \bigcup_{B \in [Br]_A} B$ By Corollary 1 of Proposition 8, we know that there exists a branch $B \in [Br]_A$, for which M is a model of $\gamma(\text{slice}(\phi, B))$. □