Reducing CTL-live Model Checking to First-Order Logic Validity Checking

Amirhossein Vakili and Nancy A. Day
Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada, N2L 3G1
{avakili,nday}@uwaterloo.ca

***Abstract*—Temporal logic model checking of infinite state systems without the use of iteration or abstraction is usually considered beyond the realm of first-order logic (FOL) reasoners because of the need for a fixpoint computation. In this paper, we show that it is possible to reduce model checking of a finite or infinite Kripke structure that is expressed in FOL to a validity problem in FOL for a fragment of computational tree logic (CTL), which we call *CTL-live*. CTL-live includes the CTL connectives that are traditionally used to express liveness properties. Our reduction can form the basis for methods that use FOL reasoning techniques directly to accomplish model checking of CTL-live properties without the need for fixpoint operators, transitive closure, abstraction, or induction.**

I. Introduction

Model checking is the problem of checking whether a Kripke structure satisfies a temporal logic formula [1]. Model checking has been used extensively to verify and find bugs in finite state systems. To deal with the growing complexity of software and hardware systems, we need methods that can analyze more abstract models so that we can discover errors earlier in the development process. The progress in SMT (satisfiability modulo theories) solvers [2] has turned first-order reasoners into powerful, efficient verification tools. In this paper, we examine the challenge of using first-order logic (FOL) to express the temporal logic model checking problem for models described in FOL.

Existing model checking methods that use first-order reasoners can be divided into two major categories: 1) bounded model checking (e.g., [3], [4]) and 2) unbounded model checking (e.g., [5], [6]). Bounded methods check whether a property holds for a certain length of execution path by creating a formula consisting of the transition relation expanded to the desired bound. Since the bound is finite, the problem can be expressed in FOL, therefore, FOL reasoners can be used to solve the entire bounded (and therefore incomplete) model checking problem at one time. Unbounded methods call a FOL reasoner multiple times iteratively to traverse the reachable state space. This iteration can result in parts of the reasoning being redone multiple times. These methods are mostly used for safety properties; for infinite systems, termination (without approximation) is guaranteed only in the case where the property is violated. FOL reasoners, such as SMT solvers, have not been used to solve an entire unbounded model checking problem in one call because model checking is a question of reachability within a graph (in this case a Kripke structure), and the reachability relation (transitive closure) is not expressible in FOL. Therefore, temporal logic model checking for infinite state systems without the use of iteration or abstraction is usually considered beyond the realm of FOL reasoners.

Our contribution is to show that model checking an interesting fragment of computational tree logic (CTL) [7], which we call *CTL-live*, is reducible to validity checking in FOL; in other words, model checking a CTL-live property of a Kripke structure can be done completely using deductive techniques of FOL. Thus, some reachability queries can be answered using a FOL reasoner even though the reachability relation itself is not expressible in FOL. CTL-live includes the CTL connectives that are often used to express liveness properties (e.g., **AF**, **AU**, etc.). Our result holds for any Kripke structure expressible symbolically in FOL. Since FOL validity checking is recursively enumerable (r.e.) [8], if a Kripke structure satisfies a CTL-live property our reduction can be used to generate a proof automatically. This is the opposite of iterative unbounded methods, such as [6], which guarantee termination only if the property is not satisfied.

Model checking a CTL formula φ requires checking whether the set of initial states of a Kripke structure is included in the set of states that satisfy φ. Validity in FOL is defined using a universal quantifier over interpretations, which is not a first-order quantifier. The key insight in our approach is to use this implicit higher-order quantifier to quantify over sets that include every state that satisfies φ and possibly more; these sets along with this higher-order quantifier are sufficient to solve the model checking problem for a CTL-live formula.

Our result can form the basis for using first-order reasoners directly for model checking CTL-live properties of infinite Kripke structures expressed symbolically in FOL. By avoiding external iteration, we allow the reasoning tool to work at its maximum efficiency with respect to reusing parts of the deduction. By avoiding manual abstraction, we have removed a large burden on the user to justify the validity of the abstraction.

II. Preliminaries

We use standard first-order logic with equality (FOL) [8]. The syntax and semantics of FOL is defined using the concepts

ISBN: 978-0-9835678-4-4.

of signatures and interpretations. A *signature* is a set of *functional* and *relational symbols* where each symbol has a corresponding *arity*, which is a natural number. For a given signature, an *interpretation* consists of a *domain* (a non-empty set), and a *mapping*, which determines the content of each functional and relational symbol in the signature. We use the notation $X^{\mathcal{I}}$ to denote the value that the symbol X is mapped to under the interpretation $\mathcal{I}$.

We denote the *satisfiability* relation for FOL by $\Vdash$, where $\mathcal{I} \Vdash \Phi$ means that the interpretation $\mathcal{I}$ *satisfies* the FOL formula Φ, and $\mathcal{I} \nVdash \Phi$ denotes otherwise. If Γ is a set of FOL formulae and $\mathcal{I}$ is an interpretation, the notation $\mathcal{I} \Vdash \Gamma$ means that $\mathcal{I}$ satisfies every formula in Γ. *Validity* (or semantic entailment) in FOL is denoted by $\Gamma \models \Phi$.

The subset relation symbol ($\subseteq$) is overloaded in this paper: suppose X and Y are relational symbols with arity 1; the formula $X \subseteq Y$ is a short form for $\forall s: \ X(s) \rightarrow Y(s)$.

Computational tree logic (CTL) is a temporal logic to specify properties over time [7]. A temporal connective of CTL consists of two parts: a path and a state quantifier. A path quantifier is either **E** (there exists a path) or **A** (for all paths). The state quantifiers are **X** (next state), **F** (eventually), **G** (globally), and **U** (strong until). The semantics of CTL formulae is defined using Kripke structures. A *Kripke structure* is a four tuple, $\mathcal{K} = \langle \mathcal{S}, \mathcal{S}_0, \mathcal{N}, \mathbb{P} \rangle$, where: $\mathcal{S}$ is a set of states; $\mathcal{S}_0$, the set of initial states, is a non-empty subset of $\mathcal{S}$; $\mathcal{N}$, the next-state relation, is a total binary relation over $\mathcal{S}$; $\mathbb{P}$ is a finite set of unary predicates over states. Predicates represent the local properties of the states, and are called *labelling predicates*.

The notation $\mathcal{K}, s \Vdash_c \varphi$ denotes that the state s of the Kripke structure $\mathcal{K}$ satisfies the CTL formula φ and $\mathcal{K}, s \nVdash_c \varphi$ denotes otherwise. We use the standard semantics of CTL [1].

The set of states of a Kripke structure $\mathcal{K}$ that satisfies a CTL formula φ is denoted by $[\varphi]_{\mathcal{K}}$:

$$[\varphi]_{\mathcal{K}} = \{s \in \mathcal{S} \mid \quad \mathcal{K}, s \Vdash_c \varphi\}$$

The Kripke structure $\mathcal{K}$ satisfies the CTL formula φ, denoted by $\mathcal{K} \Vdash_c \varphi$, iff for all $s \in \mathcal{S}_0$ we have $\mathcal{K}, s \Vdash_c \varphi$:

$$\mathcal{K} \Vdash_c \varphi \iff \mathcal{S}_0 \subseteq [\varphi]_{\mathcal{K}}$$

III. Overview

The goal of this work is to reduce the model checking problem ($\Vdash_c$) to validity checking in FOL ($\models$). The first step is to represent a Kripke structure symbolically in FOL. For a Kripke structure $\mathcal{K} = \langle \mathcal{S}, \mathcal{S}_0, \mathcal{N}, \mathbb{P} \rangle$, its symbolic representation (*symbolic*(K)) is a set of FOL formulae over the signature $K = \{S_0, N, P_1, .., P_n\}$ where relational symbol N has arity 2 and every other symbol has arity 1.

Since *symbolic*(K) is a set of FOL formulae, it can have multiple satisfying interpretations (each of which is a Kripke structure) that are not isomorphic because it may use uninterpreted functions and relations, and it may underconstrain the model.

Reduction Procedure:

INPUT:
symbolic(K) *:symbolic representation of a Kripke structure.*
φ *: a CTL-live formula.*

OUTPUT:
symbolic$(K) \bigcup$ **CTLL2FOL**$(\varphi) \models S_0 \subseteq \lceil \varphi \rceil$

Fig. 1. Reduction Procedure

We define *symbolic*$(K) \models_c \varphi$ to mean that every satisfying interpretation $\mathcal{K}$ of *symbolic*(K) satisfies the CTL formula φ:

$$\textit{symbolic}(K) \models_c \varphi \iff \forall \mathcal{K} : \mathcal{K} \Vdash \textit{symbolic}(K) \implies \mathcal{K} \Vdash_c \varphi$$

If *symbolic*(K) has only one satisfying interpretation up to isomorphism, then *symbolic*$(K) \models_c \varphi$ is equivalent to $\mathcal{K} \Vdash_c \varphi$. However, we do not require *symbolic*(K) to have only one satisfying interpretation up to isomorphism.

Our main contribution is to identify a fragment of CTL such that its model checking problem for a symbolic representation of a Kripke structure is reducible to validity checking in FOL. We call this fragment CTL-live. We show that there exists a Γ (set of FOL formulae) and Φ (FOL formula) such that:

$$\textit{symbolic}(K) \models_c \varphi \quad \iff \quad \Gamma \models \Phi$$

for φ in CTL-live. We present a function **CTLL2FOL** that takes a CTL-live φ formula as input and generates a finite set of FOL formulae that represent the satisfiability of φ. The function **CTLL2FOL** introduces a new relational symbol with arity 1 for every sub-formula of φ including φ itself. We use the notation $\lceil \varphi \rceil$ to refer to the relational symbol introduced by **CTLL2FOL** for the formula φ. The formulae generated by **CTLL2FOL** are constraints over these new relational symbols. Figure 1 is an overview of our reduction. The input of the reduction is a symbolic representation of a Kripke structure(s) (*symbolic*(K)) and a CTL-live formula (φ). The reduction procedure asserts whether the union of *symbolic*(K) with the formulae generated by **CTLL2FOL**(φ) entails $S_0 \subseteq \lceil \varphi \rceil$.

IV. Reducing CTL-live Model Checking to FOL

In this section, first, we present the intuition behind reducing model checking to FOL validity checking. Then, we define CTL-live and **CTLL2FOL**(φ).

Suppose *symbolic*(K) is a symbolic representation with a unique satisfying Kripke structure $\mathcal{K}$, and $P \in \mathbb{P}$ is a labelling predicate. We are interested in checking whether $\mathcal{K}$ satisfies **EF** P. From the semantics of CTL and its encoding in the mu-calculus [1], we know that the set of states that satisfy **EF** P, $[\textbf{EF } P]_{\mathcal{K}}$, is the **smallest** set, $\lceil \textbf{EF } P \rceil$, such that the following two FOL formulae hold:

$$\begin{array}{ll} A_1) & P \subseteq \lceil \textbf{EF } P \rceil \\ A_2) & \forall s, s' : \big(N(s, s') \wedge \lceil \textbf{EF } P \rceil(s')\big) \rightarrow \lceil \textbf{EF } P \rceil(s) \end{array} \tag{1}$$

Formula A_1 states that every state that satisfies P also satisfies **EF** P, and Formula A_2 states that if a state s has a next state that satisfies **EF** P, then s also satisfies **EF** P. We use the symbol $\lceil \textbf{EF } P \rceil$ rather than $[\textbf{EF } P]$ because there are multiple

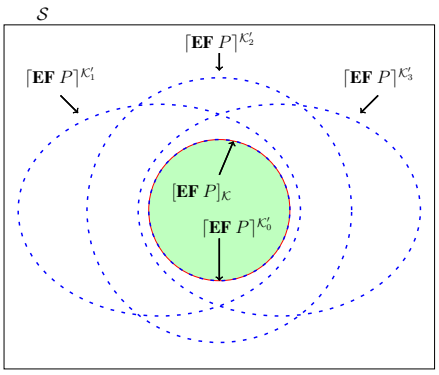

Fig. 2. Possible values for $\lceil \mathbf{EF}\ P \rceil$

sets that satisfy formulae A_1 and A_2. Any Kripke structure $\mathcal{K}'$ that is a satisfying interpretation of $symbolic(K) \bigcup \{A_1, A_2\}$ is equal to $\mathcal{K}$ plus it can map $\lceil \mathbf{EF}\ P \rceil$ to a set that includes $[\mathbf{EF}\ P]_{\mathcal{K}}$, but is potentially larger, i.e., $\lceil \mathbf{EF}\ P \rceil$ may be an overapproximation of $[\mathbf{EF}\ P]_{\mathcal{K}}$. This property is depicted in Figure 2, where $\lceil \mathbf{EF}\ P \rceil^{\mathcal{K}'_i}$ means the value of relational symbol $\lceil \mathbf{EF}\ P \rceil$ under the interpretation/Kripke structure $\mathcal{K}'_i$.

Since $[\mathbf{EF}\ P]_{\mathcal{K}}$ equals the smallest amongst the $\lceil \mathbf{EF}\ P \rceil^{\mathcal{K}'}s$ satisfying $symbolic(K) \bigcup \{A_1, A_2\}$, checking whether S_0 is a subset of $[\mathbf{EF}\ P]_{\mathcal{K}}$ is equivalent to checking whether S_0 is a subset of $\lceil \mathbf{EF}\ P \rceil^{\mathcal{K}'}$ for **every** $\mathcal{K}'$:

$$S_0 \subseteq [\mathbf{EF}\ P]_{\mathcal{K}} \iff \forall \mathcal{K}' \Vdash symbolic(K) \bigcup \{A_1, A_2\} : \mathcal{S}_0 \subseteq \lceil \mathbf{EF}\ P \rceil^{\mathcal{K}'} \quad (2)$$

The universal quantifier in Equation 2 is over interpretations, which is not available in FOL, but it is implicitly used in the definition of validity: recall that $\Gamma \models \Phi$ iff **every** satisfying interpretation of Γ satisfies Φ; therefore:

$$\mathcal{K} \Vdash_c \mathbf{EF}\ P \iff S_0 \subseteq [\mathbf{EF}\ P]_{\mathcal{K}} \iff symbolic(K) \bigcup \{A_1, A_2\} \models S_0 \subseteq \lceil \mathbf{EF}\ P \rceil$$

We reduce model checking of **EF** to validity checking in FOL by using the higher-order quantifier in the meta-language of FOL. What we have shown here is that even though the constraints on $\lceil \mathbf{EF}\ P \rceil$ in Equation 1 do not precisely express the set of states that satisfy **EF** P, they express a set that **includes** every state that satisfies **EF** P (and possibly more). Since in model checking, it is important to see whether the set of initial states is **included** in the set of states that satisfy **EF** P, these constraints along with the definition of validity in FOL, which implicitly uses a universal quantifier over interpretations, can be used to express the model checking problem for the CTL connective **EF**.

The key idea behind this result is that the CTL connective **EF** can be expressed as the smallest set that satisfies some FOL formulae. We can generalize this result for other CTL connectives that have the same property: **AF**, **EU**, and **AU**. We can also include the propositional connectives $\wedge$ and $\vee$ since their corresponding set operations (intersection and union) are monotonic with respect to set inclusion.

Temporal part		
φ	$::=$	$\pi \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2$
	$::=$	$\mathbf{EX}\varphi \mid \mathbf{AX}\varphi \mid \mathbf{EF}\varphi \mid \mathbf{AF}\varphi$
	$::=$	$\varphi_1 \mathbf{EU} \varphi_2 \mid \varphi_1 \mathbf{AU} \varphi_2$
Propositional part		
π	$::=$	$P \mid \neg\pi \mid \pi_1 \vee \pi_2$
where P is a labelling predicate.		

Fig. 3. CTL-live

Figure 3 presents the fragment of CTL for which the model checking problem can be expressed in FOL. We call this fragment *CTL-live*, since it contains the CTL connectives that are usually used to express liveness properties. CTL-live's grammar has two parts: temporal and propositional. CTL-live disallows a temporal connective to be within the scope of negation ($\neg$); e.g., the CTL formula $\neg(\mathbf{AF}\ P)$ is not part of CTL-live, but $\mathbf{AF}\ (\neg P)$ is.

To check if $symbolic(K) \models_c \varphi$ where φ is a CTL-live formula, we use a function called **`CTLL2FOL`**, shown in Figure 4, to create a set of FOL formulae expressing the meaning of these connectives. In Figure 4, $\lceil \varphi \rceil$ is a new relational symbol that is introduced by **`CTLL2FOL`** for the formula φ; for a labelling predicate P, $\lceil P \rceil$ is equal to P. The complexity of **`CTLL2FOL`** is linear with respect to the size of φ.

Theorem 1 presents our main contribution: model checking a symbolic representation of a Kripke structure(s) ($\models_c$) for a CTL-live formula is reducible to validity checking in FOL ($\models$). Complete proofs can be found in Vakili and Day [9].

Theorem 1: Let $symbolic(K)$ be a set of FOL formulae that specifies a Kripke structure(s); we have:

$$symbolic(K) \models_c \varphi \iff symbolic(K) \bigcup \mathbf{CTLL2FOL}(\varphi) \models S_0 \subseteq \lceil \varphi \rceil$$

V. Related Work

Based on [10], we reduced model checking of CTL with fairness constraints for finite symbolic Kripke structures to validity checking in FOL(TC) and used Alloy for model checking [11]. Since transitive closure for an infinite system is not expressible in FOL, this encoding cannot be used with a FOL reasoner.

K-induction is a technique for unbounded model checking of safety properties [5]. This technique extends bounded model checking by proving that bounded model checking for bound K is sufficient. The number K is dominated by the diameter of a Kripke structure. The diameter is computed iteratively using a SAT solver to check the equivalence of two formulae: the equivalence holds iff no new state can be reached by taking more than K steps. In [5], termination is guaranteed due to the finiteness of the Kripke structures under study.

ISBN: 978-0-9835678-4-4.

CTLL2FOL(φ) :

case φ of

1) P -> $\{\}$ where P is a labelling predicate
2) $\neg\psi$ -> $\{\ \forall s :\ \lceil\varphi\rceil(s) \leftrightarrow \neg\lceil\psi\rceil(s)\ \} \bigcup$ **CTLL2FOL**(ψ)
3) $\psi_1 \vee \psi_2$ -> $\{\ \forall s :\ \lceil\varphi\rceil(s) \leftrightarrow \lceil\psi_1\rceil(s) \vee \lceil\psi_2\rceil(s)\ \} \bigcup$ **CTLL2FOL**$(\psi_1) \bigcup$ **CTLL2FOL**(ψ_2)
4) $\psi_1 \wedge \psi_2$ -> $\{\ \forall s :\ \lceil\varphi\rceil(s) \leftrightarrow \lceil\psi_1\rceil(s) \wedge \lceil\psi_2\rceil(s)\ \} \bigcup$ **CTLL2FOL**$(\psi_1) \bigcup$ **CTLL2FOL**(ψ_2)
5) $\mathbf{EX}\psi$ -> $\{\ \forall s :\ \big(\exists s' :\ N(s,s') \wedge \lceil\psi\rceil(s')\big) \rightarrow \lceil\varphi\rceil(s)\ \} \bigcup$ **CTLL2FOL**(ψ)
6) $\mathbf{AX}\psi$ -> $\{\ \forall s :\ \big(\forall s' :\ N(s,s') \rightarrow \lceil\psi\rceil(s')\big) \rightarrow \lceil\varphi\rceil(s)\ \} \bigcup$ **CTLL2FOL**(ψ)
7) $\mathbf{EF}\psi$ -> $\{\ \lceil\psi\rceil \subseteq \lceil\varphi\rceil\ ,\ \ \forall s : \big(\exists s' : N(s,s') \wedge \lceil\varphi\rceil(s')\big) \rightarrow \lceil\varphi\rceil(s)\ \} \bigcup$ **CTLL2FOL**(ψ)
8) $\mathbf{AF}\psi$ -> $\{\ \lceil\psi\rceil \subseteq \lceil\varphi\rceil\ ,\ \ \forall s : \big(\forall s' : N(s,s') \rightarrow \lceil\varphi\rceil(s')\big) \rightarrow \lceil\varphi\rceil(s)\ \} \bigcup$ **CTLL2FOL**(ψ)
9) $\psi_1\mathbf{EU}\psi_2$ -> $\{\ \lceil\psi_2\rceil \subseteq \lceil\varphi\rceil\ ,\ \ \forall s : \lceil\psi_1\rceil(s) \wedge \big(\exists s' : N(s,s') \wedge \lceil\varphi\rceil(s')\big) \rightarrow \lceil\varphi\rceil(s)\ \} \bigcup$ **CTLL2FOL**$(\psi_1) \bigcup$ **CTLL2FOL**(ψ_2)
10) $\psi_1\mathbf{AU}\psi_2$ -> $\{\ \lceil\psi_2\rceil \subseteq \lceil\varphi\rceil\ ,\ \ \forall s : \lceil\psi_1\rceil(s) \wedge \big(\forall s' : N(s,s') \rightarrow \lceil\varphi\rceil(s')\big) \rightarrow \lceil\varphi\rceil(s)\ \} \bigcup$ **CTLL2FOL**$(\psi_1) \bigcup$ **CTLL2FOL**(ψ_2)

Fig. 4. Definition of **CTLL2FOL**. φ is a CTL-live formula.

Bultan, Gerber, and Pugh used Presburger formulae to represent infinite sets of states symbolically [6]. Their model checking approach requires a fixpoint calculation, and termination is achieved by using conservative approximation. This approach allows false negatives.

Kesten and Pnueli presented a sound and relatively complete (oracle based) deductive system for CTL* [12] to provide proof-like evidence for a model that satisfies a property. CTL-live is less expressive than CTL* but based on the completeness of FOL, CTL-live has a sound and complete deductive system.

Beyene, Popeea, and Rybalchenko encoded CTL model checking of infinite state systems into forall-exists quantified Horn clauses (which we call ExQH) [13]. The contribution of [13] is to develop a solver for ExQH and demonstrate its use for model checking CTL properties. Their method requires the models and the model checking constraints to be expressed in ExQH and to satisfy some well-foundedness conditions, whereas our results hold for any set of FOL constraints, which may describe multiple Kripke structures. Termination of their method is not guaranteed.

VI. Conclusion

We presented a fragment of CTL, called CTL-live, whose model checking problem is reducible to validity checking in FOL. Our reduction shows that FOL deductive techniques are sufficient for model checking CTL-live formulae, without the need for iteration, abstraction, or induction. The key insight in our approach is to use the implicit higher-order quantifier in the definition of validity to require that all initial states of a Kripke structure are within all the sets of states that satisfy an overapproximation of a CTL-live temporal operator, and thereby, reducing model checking to validity in FOL. Validity checking for FOL is r.e.; as a result, this reduction ensures that a proof can be automatically generated when a CTL-live formula is satisfied by a model. We have also proved that CTL-live is maximal in the sense that it is the largest fragment of CTL such that its model checking is reducible to FOL validity [9].

Our theory provides the basis for using first-order reasoners directly for model checking CTL-live properties of abstract and infinite Kripke structures expressed symbolically in FOL. By avoiding iteration, the tool can reuse its internal deductions to increase productivity. The rapid improvements in the efficiency of SMT solvers, FOL automated theorem proving, etc. have a direct effect on the practical application of our results. We are currently studying the use of SMT solvers for model checking CTL-live formulae [14].

References

[1] E. Clarke, O. Grumberg, and D. A. Peled, *Model Checking*. MIT Press, 1999.

[2] C. Barrett, R. Sebastiani, S. Seshia, and C. Tinelli, *Satisfiability Modulo Theories*, ser. Frontiers in Artificial Intelligence and Applications. IOS Press, February 2009, vol. 185, ch. 26, pp. 825–885.

[3] A. Biere, A. Cimatti, E. Clarke, and Y. Zhu, "Symbolic Model Checking without BDDs," in *TACAS*, ser. LNCS. Springer, 1999, pp. 193–207.

[4] T. Schüle and K. Schneider, "Bounded model checking of infinite state systems," *Formal Methods in System Design*, pp. 51–81, 2007.

[5] M. Sheeran, S. Singh, and G. Stålmarck, "Checking Safety Properties Using Induction and a SAT-Solver," in *FMCAD*, ser. LNCS. Springer, 2000, vol. 1954, pp. 127–144.

[6] T. Bultan, R. Gerber, and W. Pugh, "Symbolic Model Checking of Infinite State Systems Using Presburger Arithmetic," in *CAV*, ser. LNCS, O. Grumberg, Ed. Springer, 1997, vol. 1254, pp. 400–411.

[7] E. M. Clarke, E. A. Emerson, and A. P. Sistla, "Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications," *ACM TOPLS*, pp. 244–263, 1986.

[8] J. Harrison, *Handbook of Practical Logic and Automated Reasoning*. Cambridge University Press, 2009.

[9] A. Vakili and N. A. Day, "Reducing CTL-live Model Checking to Semantic Entailment in First-Order Logic (Version 1)," Cheriton School of Comp. Sci., University of Waterloo, Tech. Rep. CS-2014-05, 2014.

[10] N. Immerman and M. Vardi, "Model Checking and Transitive-Closure Logic," in *CAV*, ser. LNCS. Springer, 1997, vol. 1254, pp. 291–302.

[11] A. Vakili and N. Day, "Temporal Logic Model Checking in Alloy," in *ABZ*, ser. LNCS. Springer, 2012, vol. 7316, pp. 150–163.

[12] Y. Kesten and A. Pnueli, "A compositional approach to CTL* verification," *Theoretical Computer Science*, pp. 397 – 428, 2005.

[13] T. A. Beyene, C. Popeea, and A. Rybalchenko, "Solving existentially quantified horn clauses," ser. CAV. Springer, 2013, pp. 869–882.

[14] A. Vakili and N. A. Day, "Verifying CTL-live Properties of Infinite State Models using an SMT Solver," in *FSE'14*, Oct 2014, To appear.

ISBN: 978-0-9835678-4-4.