A Tool that Incrementally Approximates Finite Satisfiability in Full Interval Temporal Logic*

Rüdiger Ehlers[1,2] and Martin Lange[1]

[1]School of Electrical Engineering and Computer Science, University of Kassel, Germany
[2]University of Bremen and DFKI GmbH, Bremen, Germany

Abstract. Interval Temporal Logic (ITL) is a powerful formalism to reason about sequences of events that can occur simultaneously and in an overlapping fashion. Despite its importance for various application domains, little tool support for automated ITL reasoning is available, possibly also owed to ITL's undecidability. We consider bounded satisfiability which approximates finite satisfiability and is only NP-complete. We provide an encoding into SAT that is designed to use the power of modern incremental SAT solvers. We present a tool that tests an ITL specification for finite satisfiability.

1 Introduction

Propositional Interval Temporal Logic (ITL) [13, 9] is a modal logic that is interpreted over interval structures which enrich the natural numbers with propositional evaluations of all its intervals. Its modalities are obtained from Allen's relations on intervals [1]. Thus, it can make assertions like "every right-neighbouring interval contains an interval which ..." etc.

Despite many claims about the importance of ITL in various areas like hardware verification, A.I. planning etc., there is little tool support for automatically checking the satisfiability of an ITL formula. On one hand this may be caused by ITL's undecidability. This issue has been studied extensively together with questions regarding the expressive power and axiomatisability of ITL and its fragments, naturally obtained by restricting it to a subset of Allen's interval relations [10, 12, 11]. The complexity of their satisfiability problems varies between NP and undecidability depending on the combination of relations chosen.

There is an implementation of a tableau-based procedure [5] for the Right Neighbourhood fragment only. This is quite a weak fragment featuring a single modality only. The same fragment has also been targeted with an approach based on evolutionary algorithms [4]. This constitutes a sound but incomplete approximation method for the *finite satisfiability problem*, i.e. the question of whether or not a formula is satisfied by an interval structure based on a finite prefix of the natural numbers.

Approximative solutions can help to tackle difficult (like undecidable or just very hard) problems. One such method that is particularly successful in the area of hardware verification is *bounded model checking* [7]. It approximates a PSPACE-hard model

* The European Research Council has provided financial support under the European Community's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement no 259267.

checking problem through successive calls to a problem in NP, i.e. one that easily reduces to the satisfiability problem for propositional logic (SAT). It has been shown that such approximations can also yield useful approaches to even undecidable problems [2].

Here we report on an implementation of a similar method for the finite satisfiability problem for ITL. We show how to encode the problem of deciding whether or not a given ITL formula φ has a model of length k by a propositional formula of size polynomial in $|\varphi|$ and $|k|$. The approximation then iterates through increasing lengths k, thus being able to report satisfiability but not unsatisfiability. The encoding is *incremental*, i.e. the SAT formula for length $k + 1$ can be obtained using the SAT formula for length k as starting point and does not have to be computed from scratch. This approach bears the following advantages.

– It aims at *efficiency* by using modern SAT solvers and thus benefits from developments in this area.
– It covers the *entire* ITL unlike the few implementations described above.
– It is *extensible*; further logical operators like those of the Duration Calculus [6] or CDT [15] could easily be integrated into the encoding.

2 Interval Temporal Logic

As usual, we write $[i, j]$ when $i \leq j$ for the interval of natural numbers between i and j inclusively. For an $n \in \mathbb{N}$ let $I(n) = \{[i, j] \mid 0 \leq i \leq j < n\}$ be the set of all intervals with upper bound less than n. Let $\mathcal{P} = \{p, q, \ldots\}$ be a set of atomic propositions. A *(finite) interval structure* (over $\mathcal{P}$) is a pair $\mathcal{I} = (n, \vartheta)$ with $n \in \mathbb{N}$ and $\vartheta : I(n) \to 2^{\mathcal{P}}$. We call n the *length* of the interval structure $\mathcal{I}$.

There are twelve relations on intervals over a linear order, known as Allen's relations [1], which describe their relative position on this linear order. Here we consider four of them defined by $[i, j]\ B\ [i', j']$ iff $i = i' \wedge j' < j$ ("started-by"); $[i, j]\ E\ [i', j']$ iff $i < i' \wedge j' = j$ ("finished-by"); as well as their inverses $\bar{B}$ and $\bar{E}$ where $[i, j]\ \bar{r}\ [i', j']$ iff $[i', j']\ r\ [i, j]$.

Formulas of ITL in positive normal form over the set $\mathcal{P}$ of atomic propositions are given by the following grammar.

$$\varphi \ ::= \ p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \langle r \rangle \varphi \mid [r]\varphi$$

where $p \in \mathcal{P}$ and $r \in \{B, \bar{B}, E, \bar{E}\}$.

We use usual Boolean abbreviations like $\bot := p \wedge \neg p$ and $\top := p \vee \neg p$ for some p. Modal operators for the other eight Allen relations are definable via $\langle A \rangle \varphi := ([E]\bot \wedge \langle \bar{B} \rangle \varphi) \vee \langle E \rangle ([E]\bot \wedge \langle \bar{B} \rangle \varphi)$ ("meets"); $\langle D \rangle \varphi := \langle B \rangle \langle E \rangle \varphi$ ("contains"); $\langle L \rangle \varphi := \langle A \rangle \langle E \rangle \varphi$ ("before"); $\langle O \rangle \varphi := \langle E \rangle \langle \bar{B} \rangle \varphi$ ("overlaps") and similarly for their inverses.

The set $Sub(\varphi)$ of subformulas of φ is defined as usual. We measure the size of a formula φ in terms of the number of its different subformulas: $|\varphi| := |Sub(\varphi)|$.

An ITL formula φ is interpreted in an interval $[i, j]$ of a finite interval structure $\mathcal{I} = (n, \vartheta)$ as follows [9].

$\mathcal{I}, [i, j] \models p$	iff	$p \in \vartheta([i, j])$
$\mathcal{I}, [i, j] \models \neg p$	iff	$p \notin \vartheta([i, j])$
$\mathcal{I}, [i, j] \models \varphi \wedge \psi$	iff	$\mathcal{I}, [i, j] \models \varphi$ and $\mathcal{I}, [i, j] \models \psi$
$\mathcal{I}, [i, j] \models \varphi \vee \psi$	iff	$\mathcal{I}, [i, j] \models \varphi$ or $\mathcal{I}, [i, j] \models \psi$
$\mathcal{I}, [i, j] \models \langle r \rangle \varphi$	iff	there is $[i', j'] \in I(n)$ s.t. $[i, j] r [i', j']$ and $\mathcal{I}, [i', j'] \models \varphi$
$\mathcal{I}, [i, j] \models [r] \varphi$	iff	for all $[i', j'] \in I(n)$ with $[i, j] r [i', j']$ we have $\mathcal{I}, [i', j'] \models \varphi$

The *finite satisfiability problem* for ITL is defined as follows. Given an ITL formula φ, decide whether or not there is a finite interval structure $\mathcal{I} = (n, \vartheta)$ and an interval $[i, j] \in I(n)$ such that $\mathcal{I}, [i, j] \models \varphi$. A formula that has a model in the above sense is said to be *finitely satisfiable*.

3 Approximating Finite Satisfiability

We develop the notion of bounded ITL satisfiability which approximates finite satisfiability. To keep the presentation short we work with the minimal set of modal operators introduced above. For efficiency purposes it may be useful to treat the other operators as basic and not as abbreviations; this is done for instance in the implementation that is reported on in the next section.

Definition 1. *Let $n \geq 1$. An ITL formula φ is said to be n-bounded satisfiable if there is a finite interval structure $\mathcal{I} = (n, \vartheta)$ such that $\mathcal{I}, [0, 0] \models \varphi$. The n-bounded satisfiability problem is to decide, given an ITL formula φ, whether or not it is n-bounded satisfiable.*

First note that φ is finitely satisfiable iff there is a finite interval structure $\mathcal{I}$ such that $\mathcal{I}, [0, 0] \models \mathit{Somewhere}(\varphi)$ with $\mathit{Somewhere}(\varphi) := \varphi \vee \langle \bar{B} \rangle (\varphi \vee \langle E \rangle \varphi)$. Thus, when determining finite satisfiability it is possible to restrict the attention to satisfaction in the interval $[0, 0]$ at the cost of extending the input formula by 4 additional subformulas. Then we get that φ is finitely satisfiable iff $\mathit{Somewhere}(\varphi)$ is n-bounded satisfiable for some $n \geq 1$.

Note that n-bounded satisfiability is neither monotone nor antitone in n. Consider $\varphi_2 := \langle B \rangle \langle B \rangle \top \wedge [B][B][B] \bot$. The first conjunct is satisfied in an interval of the form $[i, j]$ when $j \geq i + 2$, the second one requires $j \leq i + 2$. Thus, it is only satisfied by intervals of length 2. Then $\langle A \rangle (\varphi_2 \wedge [\bar{B}] \bot)$ is 3-bounded satisfiable but neither 2- nor 4-bounded satisfiable. This is an important observation for the design of a procedure that successively approximates finite satisfiability by bounded satisfiability for increasing bounds. It means that one has to increase by steps of 1 for the procedure to be complete.

Let $n \geq 1$ be fixed. We reduce the n-bounded ITL satisfiability problem to the propositional satisfiability problem as follows. Given an ITL formula φ we construct a finite set $\mathcal{C}^n_\varphi := \{X^\varphi_{0,0}\} \cup \mathcal{P}^n_\varphi \cup \mathcal{T}^n_\varphi$ of propositional formulas which is satisfiable iff φ is n-bounded satisfiable. The elements of $\mathcal{P}^n_\varphi := \bigcup_{i=0}^{n-1} \bigcup_{j=i}^{n-1} \mathcal{P}^n_\varphi(i, j)$ are called the *main* constraints, and those of $\mathcal{T}^n_\varphi$ are called *temporary* constraints. This distinction is necessary to make the encoding incremental, to be explained in detail below.

$\mathcal{C}^n_\varphi$ is defined over the set of atomic propositions $\{X^\psi_{i,j}\}_{0\leq i\leq j\leq n,\psi\in Sub(\varphi)}$. Intuitively, a variable $X^\psi_{i,j}$ expresses that ψ is satisfied by $[i,j]$ in the interval structure of length n that is represented by a model of $\mathcal{C}^n_\varphi$.

Each main constraint in $\mathcal{P}^n_\varphi(i,j)$ is associated with a subformula of φ as follows.

$\neg p$	$X^{\neg p}_{i,j} \to \neg X^p_{i,j}$	$\langle E\rangle\psi$	$X^{\langle E\rangle\psi}_{i,j} \to \bigvee_{k=i+1}^{j} X^\psi_{k,j}$
$\psi_1 \wedge \psi_2$	$X^{\psi_1\wedge\psi_2}_{i,j} \to X^{\psi_1}_{i,j}$	$\langle \bar{E}\rangle\psi$	$X^{\langle \bar{E}\rangle\psi}_{i,j} \to \bigvee_{k=0}^{i-1} X^\psi_{k,j}$
	$X^{\psi_1\wedge\psi_2}_{i,j} \to X^{\psi_2}_{i,j}$	$[B]\psi$	$X^{[B]\psi}_{i,j} \to X^\psi_{i,k}, \quad k=i,\ldots,j{-}1$
$\psi_1 \vee \psi_2$	$X^{\psi_1\vee\psi_2}_{i,j} \to X^{\psi_1}_{i,j} \vee X^{\psi_2}_{i,j}$	$[\bar{B}]\psi$	$X^{[\bar{B}]\psi}_{i,j} \to X^\psi_{i,k}, \quad k=j{+}1,\ldots,n{-}1$
$\langle B\rangle\psi$	$X^{\langle B\rangle\psi}_{i,j} \to \bigvee_{k=i}^{j-1} X^\psi_{i,k}$	$[E]\psi$	$X^{[E]\psi}_{i,j} \to X^\psi_{k,j}, \quad k=i{+}1,\ldots,j$
$\langle \bar{B}\rangle\psi$	$X^{\langle \bar{B}\rangle\psi}_{i,j} \to X^\psi_{i,j+1} \vee X^{\langle \bar{B}\rangle\psi}_{i,j+1}$	$[\bar{E}]\psi$	$X^{[\bar{E}]\psi}_{i,j} \to X^\psi_{k,j}, \quad k=0,\ldots,i{-}1$

Each of them is defined by case distinction on the type of the corresponding subformula. For every subformula of the form in the left column, $\mathcal{P}^n_\varphi(i,j)$ contains *all* the constraints given in the corresponding right column.

The temporary constraints in $\mathcal{T}^n_\varphi$ are defined to be $\{\neg X^{\langle \bar{B}\rangle\psi}_{i,n} \mid 0 \leq i < n, \langle \bar{B}\rangle\psi \in Sub(\varphi)\}$. Note that these variables describe the truth value of subformulas on intervals that do not exist with the currently considered length of an interval structure. They are used in the permanent constraints in order to make the encoding for the $\langle \bar{B}\rangle$-operators incremental. Since these intervals do not exist, these variables are forced to be false by the temporary constraints. When increasing the length of considered interval structures, the temporary constraints are deleted and more permanent constraints are being used to describe the truth values of these new intervals of the form $[i,n]$ for $i \leq n$.

Theorem 2. *For all ITL formulas φ and all $n \geq 1$ we have that φ is n-bounded satisfiable iff $\mathcal{C}^n_\varphi$ is satisfiable.*

The proof is standard and therefore omitted. The following estimation on the size of $\mathcal{C}^n_\varphi$ is also easy to verify.

Lemma 3. *$\mathcal{C}^n_\varphi$ contains $\mathcal{O}(|\varphi|\cdot n^2)$ many variables and is of size $\mathcal{O}(|\varphi|\cdot n^3)$.*

Also note that $\mathcal{C}^n_\varphi$ consists of propositional clauses and can therefore be given to a SAT solver as it is. Modern SAT solvers support incremental solving meaning that, as long as in between runs of the solver only clauses are added, the next solving process can re-use information gathered in the last one like learnt clauses etc. [8]. Solvers such as `picosat` [3], which we use in the tool described here can do so even if some variable values are assumed, and satisfiability checking is only performed for assignments that respect these values. Note that all clauses in $\mathcal{T}^n_\varphi$ are single-literal clauses and can thus be used as variable value assumptions during solving.

In order to benefit from incremental SAT solving we need to explain how $\mathcal{C}^m_\varphi$ can be obtained from $\mathcal{C}^n_\varphi$ for $m > n$ adding minimal sets of clauses. Because of the remark on non-monotonicity of bounded ITL satisfiability made at the beginning of this section it makes sense to consider the case of $m = n + 1$ only. The next lemma stating the possibility of using this encoding incrementally is also easily verified. It is straightforward to extend it to the case of $m > n + 1$.

Lemma 4. *For all $n \geq 1$ we have* $\mathcal{C}_\varphi^{n+1} = (\mathcal{C}_\varphi^n \setminus \mathcal{T}_\varphi^n) \cup \big(\bigcup_{i=0}^{n} \mathcal{P}_\varphi^{n+1}(i,n) \big) \ \cup \ \mathcal{T}_\varphi^{n+1}$.

Thus, $\mathcal{C}_\varphi^{n+1}$ can be constructed from $\mathcal{C}_\varphi^n$ by removing the temporary constraints (of a cardinality that is linear in n), and then adding a quadratic number of constraints to it. This is asymptotically better than building $\mathcal{C}_\varphi^{n+1}$ from scratch which would take cubic time in n.

Based on Theorem 2 and Lemma 4 we can devise a simple approximation scheme that tests an ITL formula for finite satisfiability.

```
procedure ITLFINSATTEST($\varphi$)
    $n \leftarrow 0$
    $\mathcal{C} \leftarrow \{X_{0,0}^\varphi\}$
    repeat
        $n \leftarrow n + 1$
        $\mathcal{C} \leftarrow \mathcal{C} \cup \big( \bigcup_{i=0}^{n-1} \mathcal{P}_\varphi^n(i, n-1) \big)$
    until $\mathcal{C}$ is satisfiable assuming all assignment in $\mathcal{T}_\varphi^n$ to hold
    extract a model for $\varphi$ of size $n$ from a satisfying propositional assignment
end procedure
```

Completeness of this approximation is a direct consequence of the fact that it symbolically tests all interval structures of increasing size for being a model for its input formula. Soundness only holds in the weak sense that this method does not return any false positives. However, it is not able to detect unsatisfiability, i.e. on unsatisfiable inputs it simply does not terminate.

Theorem 5 (Completeness). ITLFINSATTEST*($Somewhere(\varphi)$) terminates on an n-bounded satisfiable φ after at most n iterations of its loop and produces a model for $Somewhere(\varphi)$.*

We remark that ITLFINSATTEST can be made to terminate on fragments of ITL for which the small model property is known. Such fragments are necessarily decidable. In such cases it suffices to run the loop up to the maximal size of a minimal model of the input formula. If none has been found, unsatisfiability can be reported.

4 Experiments

To evaluate the scalability of incremental bounded ITL satisfiability checking using the ideas described in this paper, we implemented the tool `ITLFinSat`. It is available for download at `https://github.com/progirep/ITLFinSat`. The tool is written in C++ and uses the SAT solver `picosat v.957` [3] as a library for incremental solving. The tool is completely single-threaded.

We consider three benchmark cases: an ITL model of the Fischer Mutex protocol [14], a formalisation of a binary counter, and a classical puzzle as an ITL satisfiability problem.

Table 1. Results of the experiments.

benchmark φ	$\lvert\varphi\rvert$	prop. formula size	# prop. variables	model size k	time
Fischer, $n = 2$	414	45,441	31,482	10	0.18s
Fischer, $n = 3$	638	103,987	66,885	12	0.72s
Fischer, $n = 4$	880	201,650	121,800	14	6.2s
Fischer, $n = 5$	1140	352,529	201,501	16	311s
chicken puzzle	215	19,705	15,840	9	0.09s
5-bit counter	195	77,028	40,869	17	0.3s
6-bit counter	236	464,168	171,955	33	3.2s
7-bit counter	277	3,178,956	749,529	65	58.9s
8-bit counter	318	24,087,440	3,312,335	129	31m57s

Fischer protocol. This protocol orchestrates n agents that want to enter some critical section. Mutual exclusion is achieved through a clever set-wait-and-test phase in which each agent can indicate their intention to enter by setting a common variable to its ID, then wait for a while and enter the critical section only if the variable's value still equals the agent's ID. We formalise the possibility for more than one agent to enter their critical section at the same time as an ITL satisfiability problem, using 4 propositions for each agent to indicate the state that they are in currently, and $n + 1$ propositions for the common variable's values. The ITL formula then expresses that at every time the agents' states and the variable's value are unique, and that the agents can only change states according to the description above, i.e. when the variable's value allows them to do so.

Intervals in a model for this formalisation can be seen as durations for how long the agents need to remain in certain states, and the satisfiability check reveals that mutual exclusion does not hold when the waiting phase is too short for some agents. Note that correctness of Fischer's protocol relies on some phases being longer than others, and general interval length comparisons are not formalisable in plain ITL. Mutual exclusion in this protocol does depend on certain intervals being longer than others, though; this is why the reason for violation in this example would have to be found manually from the output of the satisfiability test. For the Fischer protocol, we model the question if for some value of n if for a setting with n processes, all of them can be in their critical regions at the same time as an ITL formula.

The Chicken Crossing Puzzle. We formulate the classical problem of the farmer trying to get a fox, a chicken and some corn across the river without ever leaving the chicken with the fox or the corn unattended on one side. The existence of a solution can naturally be formalised in ITL using propositions for the locations (i.e. side of the river) of the four protagonists. The ITL formula states that none of them is on both sides at the same time, that the farmer can only take one of them across the river at a time, that they are all on the left side at the beginning and on the right side at the end, etc.

Binary counter. This benchmark family is used to test the limits of the SAT-based approach. It formalises the evolution of an n-bit counter using propositions for "the i-th bit is set/unset on this interval" by stating that the highest bit is unset and afterwards set, and whenever bit i is set or unset on an interval then this begins with an interval in which bit $i-1$ is unset and ends in one in which bit $i-1$ is set. Moreover, we require that phases in which some bit is set resp. unset must not overlap. This formula for n bits is always satisfiable, but its shortest models are of length $2^{n-1}+1$.

Table 1 presents data collected from satisfiability checks for these benchmarks. All experiments were carried out on a computer with an Intel i5-3230M CPU running at 2.60GHz. A memory limit of 2GB was never exceeded in our experiments. The table shows the size of the underlying ITL formula, the size and number of propositional variables of its SAT encoding when it has been found to be k-bounded satisfiable, the size k of the model that has been found, and the overall time taken for the satisfiability check including the encoding and checks at model sizes less than k.

References

1. J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.
2. R. Axelsson, K. Heljanko, and M. Lange. Analyzing context-free grammars using an incremental SAT solver. In *ICALP'08*, volume 5126 of *LNCS*, pages 410–422. Springer, 2008.
3. Armin Biere. Picosat essentials. *JSAT*, 4(2-4):75–97, 2008.
4. D. Bresolin, F. Jiménez, G. Sánchez, and G. Sciavicco. Finite satisfiability of propositional interval logic formulas with multi-objective evolutionary algorithms. In *FOGA'13*, pages 25–36. ACM, 2013.
5. D. Bresolin, D. Della Monica, A. Montanari, and G. Sciavicco. A tableau system for right propositional neighborhood logic over finite linear orders: An implementation. In *TABLEAUX'13*, volume 8123 of *LNCS*, pages 74–80. Springer, 2013.
6. Z. Chaochen, C. A. R. Hoare, and A. P. Ravn. A calculus of durations. *Information Processing Letters*, 40(5):269–276, 1991.
7. E. M. Clarke, A. Biere, R. Raimi, and Y. Zhu. Bounded model checking using satisfiability solving. *Formal Methods in System Design*, 19(1):7–34, 2001.
8. N. Eén and N. Sörensson. An extensible SAT-solver. In *SAT'03*, volume 2919 of *LNCS*, pages 502–518. Springer, 2003.
9. J. Y. Halpern and Y. Shoham. A propositional modal logic of time intervals. In *LICS'86*, pages 279–292. IEEE, 1986.
10. I. M. Hodkinson, A. Montanari, and G. Sciavicco. Non-finite axiomatizability and undecidability of interval temporal logics with C, D, and T. In *Proc. 22nd Conf. on Computer Science Logic, CSL'08*, volume 5213 of *LNCS*, pages 308–322. Springer, 2008.
11. D. Della Monica, V. Goranko, A. Montanari, and G. Sciavicco. Expressiveness of the interval logics of allen's relations on the class of all linear orders: Complete classification. In *IJCAI'11*, pages 845–850. AAAI, 2011.
12. D. Della Monica, V. Goranko, A. Montanari, and G. Sciavicco. Interval temporal logics: a journey. *Bulletin of the EATCS*, 105, 2011.
13. B. Moszkowski. *Reasoning about digital circuits*. Ph.D. thesis, Stanford Univ., 1983.
14. G. L. Peterson and M. J. Fischer. Economical solutions to the critical section problem in a distributed system. In *STOC'77*, pages 91–97. ACM, 1977.
15. Y. Venema. A modal logic for chopping intervals. *Journal of Logic and Computation*, 1(4):453–476, September 1991.