A Tableau System for Right Propositional Neighborhood Logic over Finite Linear Orders: an Implementation *

D. Bresolin[1], D. Della Monica[2], A. Montanari[3], and G. Sciavicco[4]

[1] Department of Computer Science, University of Verona
Verona, Italy (davide.bresolin@univr.it)
[2] School of Computer Science, University of Reykjavik
Reykjavik, Iceland (dariodm@ru.is)
[3] Department of Mathematics and Computer Science, University of Udine
Udine, Italy (angelo.montanari@dimi.uniud.it)
[4] Department of Information, Engineering and Communications
University of Murcia, Murcia, Spain(guido@um.es)

Abstract. Interval temporal logics are quite expressive temporal logics, which turn out to be difficult to deal with in many respects. Even finite satisfiability of simple interval temporal logics presents non-trivial technical issues when it comes to the implementation of efficient tableau-based decision procedures. We focus our attention on the logic of Allen's relation *meets*, a.k.a. Right Propositional Neighborhood Logic (RPNL), interpreted over finite linear orders. Starting from a high-level description of a tableau system, we developed a first working implementation of a decision procedure for RPNL, and we made it accessible from the web. We report and analyze the outcomes of some initial tests.

1 Introduction

Propositional interval temporal logics play a significant role in computer science, as they provide a natural framework for representing and reasoning about temporal properties in a number of application domains [10]. Interval logic modalities correspond to relations between (pairs of) intervals. In particular, Halpern and Shoham's modal logic of time intervals HS [11] features a set of modalities that make it possible to express all Allen's interval relations [1]. HS turns out to be undecidable over all meaningful classes of linear orders, including the class of finite linear orders we are interested in. Temporal reasoning on finite linear orders

* The authors acknowledge the support from the Spanish fellowship program *'Ramon y Cajal' RYC-2011-07821* and the Spanish MEC project *TIN2009-14372-C03-01* (G. Sciavicco), the project *Processes and Modal Logics* (project nr. 100048021) of the Icelandic Research Fund and the project *Decidability and Expressiveness for Interval Temporal Logics* (project nr. 130802-051) of the Icelandic Research Fund in partnership with the European Commission Framework 7 Programme (People) under "Marie Curie Actions" (D. Della Monica), and the Italian GNCS project *Logiche di Gioco estese* (D. Bresolin and A. Montanari).

comes into play in a variety of areas. This is the case, for instance, with planning problems, which consist of finding a finite sequence of actions that, applied to an initial state of the world, leads to a goal state within a bounded amount of time, satisfying suitable conditions about which sequence of states the world must go through. In the last years, a lot of work has been done on (un)decidability and complexity of HS fragments. The complete picture about finite linear orders is given in [6]: there exist 62 non-equivalent decidable fragments of HS, partitioned into four complexities classes, ranging from NP-complete to non-primitive recursive. For each decidable fragment, an optimal decision procedure has been devised. Nevertheless, none of them is available as a working system (they are declarative procedures, which turn out to be unfeasible in practice), with the only exception of the logic of subintervals D over dense linear orders [3]. D has been implemented in LoTrec [8], a generic theorem prover that allows one to specify the rules for his/her own modal/temporal logic. Unfortunately, in general LoTrec is not suitable for interval logics (D over dense linear orders is a very special case), because: (i) it does not support the management of world labels explicitly, and (ii) it does not allow closing conditions based on the number of worlds generated in the construction of a tentative model, but only closing conditions based on patterns and repetitions.

In this paper, we focus our attention on one of the simplest decidable fragment of HS, namely, Right Propositional Neighborhood Logic (RPNL) [4,9], interpreted on finite linear orders, whose satisfiability problem has been proved to be NEXPTIME-complete. RPNL features a single modality corresponding to Allen's relation *meets*. We devised and implemented a working tableau-based decision procedure for RPNL, based on the original (non-terminating) tableau system given in [9], which exploits the small model theorem proved in [7] to guarantee termination.

2 Syntax and semantics of RPNL

Let $\mathbb{D} = \langle D, < \rangle$ be a finite linear order. An *interval* over $\mathbb{D}$ is an ordered pair $[x, y]$, with $x, y \in D$ and $x < y$ (*strict semantics*). Formulas of RPNL are obtained from a countable set $\mathcal{AP}$ of proposition letters using the standard boolean connectives $\vee$, $\wedge$ and $\neg$, and the temporal modalities $\langle A \rangle$ and $[A]$ (defined as a shorthand for $\neg\langle A\rangle\neg$). Formulas are interpreted on models $M = \langle \mathbb{D}, V \rangle$, where $V : \mathbb{I}(\mathbb{D}) \to 2^{\mathcal{AP}}$ is a *valuation* function that associates every interval of $\mathbb{D}$ with the set of proposition letters that hold true on it. The satisfiability relation $\Vdash$ is defined by the semantic clauses for propositional logic plus the modal clause

$$M, [x, y] \Vdash \langle A \rangle \text{ iff there exists } z > y \text{ such that } M, [y, z] \Vdash \varphi.$$

As shown in [7], satisfiability of RPNL-formulas can be reduced to *initial* satisfiability, that is, satisfiability on the interval $[0, 1]$. Hence, it holds that an RPNL-formula φ is satisfiable if and only if there is a model M such that $M, [0, 1] \Vdash \varphi$.

The decidability proof given in [7] shows that any RPNL-formula φ is satisfiable over finite linear orders if and only if it is satisfiable over a finite linear

order whose domain has cardinality strictly less than $2^m \cdot m + 1$, where m is the number of diamonds and boxes in φ. This provides a termination condition that can be used to implement a *fair* procedure that exhaustively searches for a model of size smaller than or equal to the bound. In this paper, we develop and implement a tableau-based decision procedure for RPNL by tailoring the general algorithm described in [9] to it and making use of the bound on the size of the model to guarantee completeness.

3 The tableau system for RPNL

The abstract structure of a tableau for RPNL is a *rooted tree* where each node is labeled with an *annotated formula* of the form $\psi : [x, y]$, which states that ψ holds over the interval $[x, y]$ on D. Every branch B of the tableau is associated with a finite domain $D_B = \{x_0, x_1, \ldots, x_N\}$ and it represents a partial model for the input formula. At each step of tableau construction, a branch and a node on it are selected and one of the *expansion rules* is applied to expand the branch. Expansion rules follow the semantics of RPNL. They include classical propositional rules plus two additional rules for modalities $[A]$ and $\langle A \rangle$:

$$(box)\ \frac{[A]\psi : [x_i, x_j]}{\psi : [x_j, x_{j+1}], \ldots, \psi : [x_j, x_N]},$$

$$(dia)\ \frac{\langle A \rangle \psi : [x_i, x_j]}{\psi : [x_j, x_{j+1}] \mid \ldots \mid \psi : [x_j, x_N] \mid \psi : [x_j, x'_j] \mid \ldots \mid \psi : [x_j, x'_N]},$$

where, for each $j \leq h \leq N$, x_h is a point in D_B and x'_h is a new point added to D_B and placed immediately after x_h and immediately before x_{h+1} (when $h < N$). The *(dia)* rule explores all possible ways of satisfying the formula ψ: either it satisfies ψ on an existing interval (nodes labelled with $\psi : [x_j, x_h]$) or it adds a new point x'_h to the domain and it satisfies ψ on the new interval $[x_j, x'_h]$. Similarly, the *(box)* rule asserts that ψ must be true on every existing interval starting at x_j. Thus, the point x_i never appears in the consequent of the rules. A branch in the tableau is declared *closed* if either $p : [x_i, x_j]$ and $\neg p : [x_i, x_j]$ occur on the branch, for some $p \in \mathcal{AP}$ and interval $[x_i, x_j]$ (*contradictory branch*); or the cardinality of the domain D_B is greater than the upper bound on the size of models (*too long branch*). Otherwise, it is considered *open*. Expansion rules are applied only to open branches (closed branches are discarded).

Given a branch B, an annotated formula $\psi : [x_i, x_j]$ is said to be *inactive on* B if and only if ψ is a literal or the rule for ψ has been already applied to it on B, it is *active on* B otherwise. The branch-expansion strategy applied by the system is the simplest possible one: the first (top-down) active formula of the current branch is selected, expanded, and deactivated. Whenever an open branch with no active formulas is found, the procedure terminates with success (the formula is satisfiable). If all branches are closed, the procedure terminates with failure (the formula is unsatisfiable).

4 Implementation of the tableau system for RPNL

In this section, we illustrate the difficulties we encountered and the implementation choices we made to turn the tableau system described in Section 3 into a computer program. The code of our implementation is written in C++ and it makes no use of external libraries, except for the C++ Standard Library. We exploited suitable data structures to represent formulas, nodes, and branches of the tableau, and we developed a search procedure that keeps track of currently-open branches and expands them by applying expansion rules according to the expansion strategy.

Representation of formulas, nodes, and branches. Since in most applications the input formula φ encodes a set of requirements to be jointly satisfied, e.g., those of a plan, we assume φ to be a logical conjunction, whose conjuncts are entered as distinct lines of a text file. φ is first transformed into an equivalent formula in negated normal form $nnf(\varphi)$. Since such a transformation does not change the number of diamonds and boxes, it does not affect the bound on the maximum cardinality of the domain. Then, φ is stored as a *syntactic tree*, whose leaves are labeled with proposition letters and whose internal nodes are labeled with Boolean connectives and modalities. In such a way, each subformula of φ corresponds to a subtree of the syntactic tree. Nodes of the tableau are represented by a structure with four components: a pointer to the subtree representing the formula labeling the node, two integer variables x and y, that identify the interval annotating the formula, and a Boolean flag, which specifies whether the node is active or not. A branch B is implemented as a list of nodes, enriched with two integer variables N and A representing respectively the cardinality of the domain D_B and the number of active nodes.

The search procedure. The search procedure stores the open branches to be expanded into a *priority queue*. At the beginning, the queue contains only the single node initial branch $\{\varphi : [0, 1]\}$. Then, the procedure operates as follows: 1. it extracts the branch B with the highest priority from the queue; 2. it checks whether B meets the closure conditions; if so, it deletes the branch and it restarts from 1; 3. it finds the closest-to-the-root active node ν in B; if there are no active nodes in B, it terminates with success and it returns B as a model for φ; 4. it applies the appropriate expansion rule to ν, it deactivates ν, it inserts the branches created by the rule into the queue, and it restarts from 1. The expansion loop is repeated until either a model for φ is found or the queue becomes empty. In the latter case, no model for φ can be found, and the formula is declared unsatisfiable.

Priority policies. The priority policy of the queue determines the next branch to expand. We implemented five different policies: i) the standard *FIFO* (First In, First Out) policy; ii) expand the branches with the *smallest domain* first (SDF); iii) expand the branches with the *largest domain* first (LDF); iv) expand the branches with the *smallest number of active nodes* first (SAN); v) expand the branches with the *greatest number of active nodes* first (GAN). All the policies are complete: they will eventually check every possible model for the input

formula with cardinality less than or equal to the selected bound. By default, the queue follows the FIFO policy, but the user can easily opt for a different one for a particular problem.

Branch expansion. If the current branch B (extracted from the queue) is declared open at step 2 of the search procedure, nodes in B are scanned to determine the closest-to-the-root active node ν. The expansion of B depends on the shape of the formula labeling ν. Three cases are possible.

Boolean formula. Since formulas are assumed to be in negated normal form, the only possible rules are the $\vee$-rule and the $\wedge$-rule. Let ν be labeled with $\psi \vee \tau : [x_i, x_j]$ (the case $\psi \wedge \tau : [x_i, x_j]$ is similar and thus omitted). We must distinguish four scenarios: (i) both $\psi : [x_i, x_j]$ and $\tau : [x_i, x_j]$ are already on B, (ii) $\psi : [x_i, x_j]$ is on B, while $\tau : [x_i, x_j]$ is not, (iii) $\tau : [x_i, x_j]$ is on B, while $\psi : [x_i, x_j]$ is not, and (iv) neither of the two is on B. In case (i), there is no need to apply the rule: $\psi \vee \tau : [x_i, x_j]$ is deactivated and B is put back in the queue. In case (ii), a copy of the branch is generated and the annotated formula $\tau : [x_i, x_j]$ is added to it; then, $\psi \vee \tau : [x_i, x_j]$ is deactivared and both B and its copy are added to the queue. Case (iii) is completely symmetric. In case (iv), two copies of the branch are generated: one is expanded with the annotated formula $\psi : [x_i, x_j]$, the other one with $\tau : [x_i, x_j]$. Then, $\psi \vee \tau : [x_i, x_j]$ is deactivated, both copies of B are added to the queue, and the original B is discarded.

Box formula $[A]\psi : [x_i, x_j]$. The box rule is applied. First, we deactivate the formula $[A]\psi : [x_i, x_j]$; then, for each $x_j < x_h \leq x_N$, if $\psi : [x_j, x_h]$ does not belong to B, we add it; finally, the expansion of B is inserted into the queue.

Diamond formula $\langle A\rangle\psi : [x_i, x_j]$. The diamond rule is applied. First, we check whether for some $x_h > x_j$ the annotated formula $\psi : [x_j, x_h]$ is on B. If this is the case, we deactivate $\langle A\rangle\psi : [x_i, x_j]$ and we put B back in the queue. Otherwise, we create a distinct copy of B for every possible way of satisfying ψ: $N - j$ copies $B_{j+1}, \ldots, B_N$, with domain cardinality N, that will be expanded with the annotated formulas $\psi : [x_j, x_{j+1}], \ldots, \psi : [x_j, x_N]$, respectively; $N - j + 1$ copies $B'_j, \ldots, B'_N$, with domain cardinality $N + 1$, that will be expanded with the annotated formulas $\psi : [x_j, x'_j], \ldots, \psi : [x_j, x'_N]$, respectively. For each copy B'_h, the expansion of the domain is obtained as follows: (i) every annotated formula $\tau : [x_k, x_l]$ such that $x_k > x_h$ is replaced by the annotated formula $\tau : [x_k + 1, x_l + 1]$. If $x_k \leq x_h < x_l$, the annotated formula is replaced by $\tau : [x_k, x_l + 1]$, while if $x_l \leq x_h$, the annotated formula remains unchanged; (ii) we add a new node labeled with the annotated formula $\psi : [x_j, x_h+1]$; (iii) we reactivate all annotated formulas $[A]\tau : [x_k, x_l]$ with $x_l \leq x_h$. To conclude the expansion, we deactivate $\langle A\rangle\psi : [x_i, x_j]$, we put all $2 \cdot (N - j) + 1$ copies of B in the queue, and we discard B.

5 Experiments

We have tested our implementation against a benchmark of different problems, divided into two classes. First, we tested the scalability of the program with

COMBINATORICS

n	Policy (sec) FIFO	SDF	LDF	SAN	GAN	Outcome (size)	n	Policy (sec) FIFO	SDF	LDF	SAN	GAN	Outcome (size)
1	0.004	0.004	0.004	0.004	0.004	4	12	1.67	–	–	1.79	–	15
2	0.004	0.008	0.004	0.004	0.008	5	13	2.73	–	–	2.94	–	16
3	0.008	0.15	0.03	0.008	0.03	6	14	4.25	–	–	4.55	–	17
4	0.01	–	30.07	0.01	30.29	7	15	6.56	–	–	7.08	–	18
5	0.012	–	–	0.012	–	8	16	9.77	–	–	10.82	–	19
6	0.02	–	–	0.03	–	9	17	14.42	–	–	15.40	–	20
7	0.07	–	–	0.07	–	10	18	20.79	–	–	22.20	–	21
8	0.15	–	–	0.16	–	11	19	29.28	–	–	32.11	–	22
9	0.3	–	–	0.32	–	12	20	40.91	–	–	44.09	–	23
10	0.56	–	–	0.59	–	13	21	–	–	–	–	–	–
11	0.99	–	–	1.06	–	14	22	–	–	–	–	–	–

RANDOMIZED

n	Policy (sec) FIFO	SDF	LDF	SAN	GAN	Outcome (size)	n	Policy (sec) FIFO	SDF	LDF	SAN	GAN	Outcome (size)
1	0.004	0.004	0.004	0.004	0.004	4	19	1.66	45.43	0.68	1.91	0.02	3 / 4
2	0.004	0.004	0.004	0.004	0.004	4	20	0.02	0.004	0.03	0.03	0.004	2 / 4
3	0.004	0.004	0.004	0.004	0.004	4	21	0.004	0.004	0.004	0.004	0.004	4
4	0.004	0.004	0.004	0.004	0.004	4	22	0.74	14.08	0.004	1.04	0.004	4
5	0.004	0.004	0.004	0.004	0.004	4	23	0.004	0.004	0.004	0.004	0.004	4
6	0.004	0.004	0.004	0.004	0.004	4	24	0.004	0.004	0.004	0.004	0.004	4
7	0.07	0.23	0.004	0.18	0.004	3 / 4	25	–	–	–	–	–	–
8	0.004	0.004	0.004	0.004	0.004	4	26	0.004	0.004	0.004	0.004	0.004	4
9	0.004	0.004	0.004	0.004	0.004	4	27	0.004	–	0.004	0.01	–	3 / 4
10	0.004	0.004	0.004	0.004	0.004	4	28	0.004	0.004	0.004	0.004	0.004	4
11	0.004	0.004	0.004	0.004	0.004	4	29	0.004	–	0.004	0.004	0.004	4
12	0.004	0.004	0.004	0.004	0.004	4	30	0.14	0.08	0.04	0.19	0.01	2 / 4
13	0.01	0.04	0.004	0.02	0.004	4	31	0.004	0.004	0.004	0.004	0.004	unsat
14	0.004	0.004	0.004	0.004	0.004	4	32	0.25	–	0.02	0.31	0.004	2 / 4
15	0.004	0.004	0.004	0.004	0.004	4	33	0.004	0.004	0.004	0.004	0.004	4
16	0.004	1.37	0.004	0.01	0.004	4	34	–	–	0.02	0.004	0.02	2 / 4
17	0.004	0.004	0.004	0.004	0.004	4	35	0.004	–	0.004	–	0.004	2 / 4
18	0.004	0.004	0.004	0.004	0.004	3	36	–	–	–	–	1.2	3

Table 1: Experimental results

respect to a set of combinatorial problems of increasing complexity (COMBINATORICS), where the n-th combinatorial problem is defined as the problem of finding a model for the formula that contains n conjuncts, each one of the type $\langle A\rangle p_i$ $(0 \leq i \leq n)$, plus $\frac{n(n+1)}{2}$ formulas of the type $[A]\neg(p_i \wedge p_j)$ $(i \neq j)$. Then, we considered the set of 36 "easy" purely randomized formulas used in [5] to evaluate an Evolutionary Computation algorithm for RPNL finite satisfiability (RANDOMIZED). Table 1 summarizes the outcome of our experiments. For each class of problems, the corresponding table shows, for each instance n, the time necessary to solve the problem for each policy (FIFO, SDF, LDF, SAN, GAN) and the size of the obtained model (or "unsat" if the instance was proved to be unsatisfiable). A time-out of 1 minute was used to stop instances running for too long. All the experiments were executed on a notebook with an Intel Pentium Dual-Core Mobile 1.6 Ghz CPU and 2 Gb of RAM, under Ubuntu Linux 11.04. Despite being a prototypical implementation, our system runs reasonably well on the COMBINATORICS benchmark, being able to pro-

duce a result in a short time for formulas up to 20 conjuncts (and up to a model size of 23 points). The results of the RANDOMIZED benchmark allows for a first comparison with the Evolutionary algorithm in [5], and shows that the two algorithms have similar performances on the considered formulas. The tableau system was able to prove that problem 31 is unsatisfiable, while the evolutionary algorithm (being incomplete) can only provide positive answers. It is important to stress that there is no available benchmark neither for RPNL, nor for any other interval temporal logic. To overcome this limitation, we are currently working to adapt some benchmarks for the modal logic K [2] and for the temporal logic LTL [12] to the interval semantics. On the web-page *http://www.di.unisa.it/dottorandi/dario.dellamonica/tableaux/* it is possible to find the system available for testing.

References

1. J. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.
2. P. Balsiger, A. Heuerding, and S. Schwendimann. A benchmark method for the propositional modal logics K, KT, S4. *J. of Automated Reasoning*, 24(3):297–317, 2000.
3. D. Bresolin, V. Goranko, A. Montanari, and P. Sala. Tableaux for logics of subinterval structures over dense orderings. *J. of Logic and Computation*, 20(1):133–166, 2010.
4. D. Bresolin, V. Goranko, A. Montanari, and G. Sciavicco. Propositional interval neighborhood logics: Expressiveness, decidability, and undecidable extensions. *Annals of Pure and Applied Logic*, 161(3):289–304, 2009.
5. D. Bresolin, F. Jiménez, G. Sánchez, and G. Sciavicco. Finite satisfiability of propositional interval logic formulas with multi-objective evolutionary algorithms. In *Proc. of the 12th FOGA*, 2013. In press.
6. D. Bresolin, D. Della Monica, A. Montanari, P. Sala, and G. Sciavicco. Interval temporal logics over finite linear orders: the complete picture. In *Proc. of the 20th ECAI*, pages 199–204, 2012.
7. D. Bresolin, A. Montanari, and G. Sciavicco. An optimal decision procedure for Right Propositional Neighborhood Logic. *J. of Automated Reasoning*, 38(1-3):173–199, 2007.
8. L. Fariñas del Cerro, D. Fauthoux, O. Gasquet, A. Herzig, D. Longin, and F. Massacci. Lotrec: the generic tableau prover for modal and description logics. In *Proc. of the 1st IJCAR*, volume 2083 of *LNCS*, pages 453–458. Springer, 2001.
9. V. Goranko, A. Montanari, and G. Sciavicco. Propositional interval neighborhood temporal logics. *J. of Universal Computer Science*, 9(9):1137–1167, 2003.
10. V. Goranko, A. Montanari, and G. Sciavicco. A road map of interval temporal logics and duration calculi. *J. of Applied Non-Classical Logics*, 14(1–2):9–54, 2004.
11. J. Halpern and Y. Shoham. A propositional modal logic of time intervals. *J. of the ACM*, 38(4):935–962, 1991.
12. K. Y. Rozier and M. Y. Vardi. LTL satisfiability checking. *Int. J. on Software Tools for Technology Transfer*, 2(12):123–137, 2010.