SAT vs. Search for Qualitative Temporal Reasoning

Jinbo Huang[1]

Abstract. Empirical data from recent work has indicated that SAT-based solvers can outperform native search-based solvers on certain classes of problems in qualitative temporal reasoning, particularly over the Interval Algebra (IA). The present work shows that, for reasoning with IA, SAT strictly dominates search in theoretical power: (1) We present a SAT encoding of IA that simulates the use of tractable subsets in native solvers. (2) We show that the refutation of any inconsistent IA network can always be done by SAT (via our new encoding) as efficiently as by native search. (3) We exhibit a class of IA networks that provably require exponential time to refute by native search, but can be refuted by SAT in polynomial time.

1 Introduction

Qualitative temporal reasoning deals with the consistency of qualitative information about time, typically given in the form of a binary constraint network where variables represent time intervals and constraints are drawn from the Interval Algebra (IA) [1]. There are 13 base relations in IA (Figure 1), and a constraint is a union of any number of base relations so that indefinite knowledge can be represented. Consistency of IA networks is NP-complete [18].

Native search-based solvers for IA [14, 8] take advantage of the existence of a large tractable subset of the 2^{13} IA relations, known as ORD-Horn [15]: If all constraints of a network belong to ORD-Horn, then the network is consistent iff enforcement of path consistency, a polynomial-time procedure, succeeds. Given an arbitrary IA network, each constraint is partitioned into ORD-Horn relations (which can always be done as ORD-Horn includes all base relations), creating the branches of a search node. At each leaf of the search tree is then a *refinement* of the network having only ORD-Horn constraints, whose consistency can be checked quickly, and the original network is consistent iff at least one leaf gives a consistent refinement.

A SAT encoding of IA has been proposed [16], and subsequently made more compact through a form of partial path consistency [10]. Empirical data reported in the cited work indicates that certain classes of problems can be solved more efficiently by a SAT solver than by native search-based solvers. On the other hand, this encoding is based on refining every constraint into base relations, and does not utilize the tractable subset ORD-Horn. As a result, there is no direct correspondence between the behavior of the SAT solver and that of a native solver given the same problem instance, and a theoretical comparison of the two methods remains elusive.[2]

In this work, we undertake an investigation into the relative theoretical power of SAT and native search as proof systems for IA,

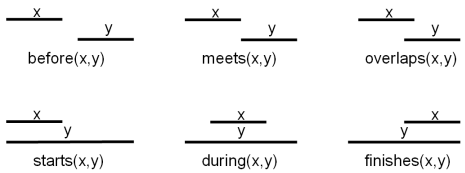

Figure 1. IA base relations: six shown above, their inverses, and the identity relation.

assuming the use of a *clause learning* [12, 13] algorithm on CNF formulas. Making use of the fact that clause learning is known to be as powerful as resolution and in particular strictly more powerful than tree-style search on CNF formulas [3, 17], we are able to establish that SAT indeed dominates native search in theoretical power for qualitative temporal reasoning with IA, as follows: (i) We present a new SAT encoding of IA based on the use of tractable subsets, where path consistency is simulated by the *unit propagation* mechanism of the SAT algorithm. (ii) With this encoding, we show that SAT can, in polynomial time, simulate native search using the same tractable subset; in other words, any inconsistent IA network can be refuted by SAT as efficiently as by native search. (iii) We cite a class of 4CNF formulas, PC_n, known as the *pebbling contradictions* [4], that have polynomial-size refutations in resolution, but not in tree-like resolution. (iv) We present a translation of 4CNF formulas into IA networks such that the 4CNF formula has a short tree-like resolution refutation in case the IA network has a short refutation by native search. This implies that native search cannot refute in polynomial time the IA networks created from PC_n. (v) We show that from our SAT encoding of the IA network, we can obtain by resolution in polynomial time a 4CNF formula isomorphic to the original 4CNF formula from which the IA network has been created. This implies that the CNF formulas encoding the IA networks created from PC_n have polynomial-size resolution refutations, and hence can be refuted by clause learning in polynomial time.

2 Background

In this paper we use the term SAT to refer to both the satisfiability problem and the approach of solving a problem by reduction to satisfiability. We assume familiarity with DPLL [7] and clause learning[3] [12, 13], two major proof systems for SAT, as well as the notion of *unit propagation*. It is known that DPLL and clause learning are respectively equivalent to tree-like resolution and (general) resolution [3, 17], and that (general) resolution is exponentially more powerful than tree-like resolution [4].

[1] NICTA and Australian National University. NICTA is funded by the Australian Government as represented by the DBCDE and the ARC through the ICT Centre of Excellence program.

[2] Other previous work on mapping qualitative constraint networks to finite-domain constraint networks and then to SAT [5, 2] likewise does not consider the use of tractable subsets and is hence not directly applicable here.

[3] While we consider restarts an inherent component of clause learning [9], some authors use "clause learning with restarts" or "conflict-directed clause learning with restarts" to refer to the same kind of proof system.

Algorithm 1 Consistency of IA networks by search

Initialization: identify the set BRANCH-POINTS of non-ORD-Horn edges of Φ
consistent(Φ)
1: enforce-path-consistency(Φ)
2: **if** Φ contains the empty relation **then**
3: **return** false
4: **if** all relations are in ORD-Horn **then**
5: **return** true
6: pick an edge (i, j) from BRANCH-POINTS not previously picked
7: split $\ell_\Phi(i, j)$ into maximal ORD-Horn relations **r**
8: **for all** $r \in$ **r do**
9: $\ell_\Phi(i, j) \leftarrow r$
10: **if** consistent(Φ) **then**
11: **return** true
12: **return** false

We now provide, in somewhat more detail, the necessary background on qualitative temporal reasoning. Given a binary constraint network Φ, we write $\mathcal{V}_\Phi$ to denote its set of variables (vertices), and ℓ_Φ the function that assigns a constraint between each pair of variables, i.e., $\ell_\Phi(i, j)$ labels the edge between variables i and j, specifying the relation between them. In particular, $\ell_\Phi(i, j)$ returns the *universal relation* (allowing all pairs of values) if no constraint is explicitly specified between the two variables.

In the case of qualitative temporal reasoning with IA, the domain of variables is the (infinite) set of all time intervals, which can be taken to be ordered pairs of rational numbers, and a constraint is one of the 2^{13} IA relations, 13 of which are *base* or *atomic* relations (see Figure 1) and the rest their unions. From here on we will abbreviate the six base relations shown in Figure 1 as b, m, o, s, d, f, their inverses as bi, mi, oi, si, di, fi, and the identity relation as eq. An IA constraint network is *consistent* if there is an assignment of a time interval to every variable such that all constraints are satisfied.

A network is *atomic* if the constraint between every pair of variables is an atomic relation. For two networks over exactly the same variables, if one is more restrictive on the relation permitted between each pair of variables, it is said to be a *refinement* of the other. Formally, for networks Φ and Ψ, Φ is a refinement of Ψ if $\mathcal{V}_\Phi = \mathcal{V}_\Psi$ and $\ell_\Phi(i, j) \subseteq \ell_\Psi(i, j)$ for all $i, j \in \mathcal{V}_\Phi$; in particular, if Φ is also atomic, then it is an *atomic refinement* of Ψ.

Path consistency is a central tool in qualitative temporal (and spatial) reasoning. A network Φ is *path consistent* if any instantiation of two variables that satisfies the constraint between them can be extended to any other variable such that all three constraints on the triangle are satisfied. More formally, we have

Definition 1 *A binary constraint network Φ is* path consistent *if for all variables $i, j, k \in \mathcal{V}_\Phi$, $\emptyset \neq \ell_\Phi(i, k) \subseteq \ell_\Phi(i, j) \circ \ell_\Phi(j, k)$, where "$\circ$" denotes the standard set-theoretic composition of two relations (i.e., $R \circ S = \{(a, c) \mid \exists b : (a, b) \in R, (b, c) \in S\}$).*

Composition of base IA relations can be obtained by looking up a known *composition table* [1]; for example, $o \circ d = \{d, o, s\}$. By convention, the set notation is taken to mean the union of the elements of the set. As composition distributes over union, composition of general relations can be obtained by composing pairs of base relations therein and taking the union of the results.

By computing compositions and intersections of relations, path consistency can be checked or enforced efficiently, in cubic time [11]. Furthermore, it is known that path consistency implies consistency for networks using only a subset of IA relations known as ORD-Horn [15], and in particular atomic IA networks [18] since ORD-Horn includes all base IA relations.

Conventional native solvers for IA [14, 8] implement a search in the space of ORD-Horn refinements of the given network, by splitting each constraint into (maximal) ORD-Horn relations and branching on them. A formulation of this procedure is given in Algorithm 1, which will be the basis for our analysis of its theoretical power as a proof system (the use of *eligible* constraints [6] has no effect on the theoretical power of the algorithm and is omitted). Note that path consistency is enforced at each search node for pruning (line 1), and the network is consistent if a path-consistent ORD-Horn refinement is found at a leaf node of the search tree (line 5). On the other hand, the network is inconsistent if at any point the empty relation is generated (as a result of taking intersections) during the enforcement of path consistency (line 3).

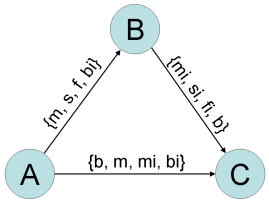

Figure 2. A triangle in an IA network.

SAT encodings of IA have been recently proposed [16, 10] where a CNF formula is constructed for a given IA network such that each model of the formula corresponds to a path-consistent atomic refinement of the network. Hence consistency of the network is reduced to satisfiability of the formula. Since these encodings do not use ORD-Horn, the behavior of a SAT solver on the CNF formulas cannot be easily related to the behavior of a native solver (i.e., an implementation of Algorithm 1) on the corresponding IA networks. Indeed, it was unknown how the two methods (SAT and native search) compare in their ultimate theoretical power. Does one intrinsically dominate the other? The rest of the paper is an investigation into this question, concluding with an answer in the affirmative.

3 New SAT Encoding Using Tractable Subsets

We start by presenting a new SAT encoding of IA such that each model of the CNF formula corresponds to a consistent ORD-Horn refinement of the network. To ensure compactness, our encoding is based on the following ideas: (i) We do not directly encode the fact that a given network has a path-consistent ORD-Horn refinement, but instead that it has an ORD-Horn refinement on which path consistency can be successfully enforced. This allows us to only encode the maximal ORD-Horn refinements of the network. (ii) We encode compositions of ORD-Horn relations by utilizing the fact that composition distributes over union. Specifically, referring to Definition 1, we encode $\ell_\Phi(i, k) \subseteq \ell_\Phi(i, j) \circ \ell_\Phi(j, k)$ by encoding the fact that "every base relation in $\ell_\Phi(i, k)$ is contained in the composition of some pair of base relations respectively from $\ell_\Phi(i, j)$ and $\ell_\Phi(j, k)$." This allows us to only encode compositions of base relations.

3.1 Variables

Our encoding of an IA network will use two groups of primary Boolean variables, referred to as *base* and *branch* variables respectively. For the triangle in Figure 2, the first group consists of the following, encoding the base relations on each edge: $AB_m, AB_s, AB_f, AB_{bi}$ (for edge AB); $BC_{mi}, BC_{si}, BC_{fi}, BC_b$ (for edge BC); $AC_b, AC_m, AC_{mi}, AC_{bi}$ (for edge AC).

The second group encodes the ORD-Horn partitions of the constraint on each edge. In this example, only edge AC has nonatomic

ORD-Horn partitions ($\{b, m\}, \{mi, bi\}$); hence the variables consist of $AC_{b,m}, AC_{mi,bi}$ (for edge AC).

In general, let $B(r)$ denote the set of all base relations contained in the relation r, and $OH(r)$ the (unique) set of (maximal) ORD-Horn partitions of r. These two groups of variables for a network Φ are: (i) IJ_p for each $p \in B(\ell_\Phi(i,j))$ and (ii) IJ_q for each $q \in OH(\ell_\Phi(i,j))\backslash B(\ell_\Phi(i,j))$, for each pair of $i, j \in \mathcal{V}_\Phi, i < j$.

A third group of variables will be used as auxiliary variables to keep the encoding compact. We will introduce these as we describe the construction of the clauses.

3.2 Clauses

We have two groups of clauses. The first "defines" the branch variables. Continuing with Figure 2, we have

$$AC_{b,m} \vee AC_{mi,bi}$$
$$AC_{b,m} \rightarrow AC_b \vee AC_m, AC_{b,m} \rightarrow \overline{AC_{mi}}, AC_{b,m} \rightarrow \overline{AC_{bi}}$$
$$AC_{mi,bi} \rightarrow AC_{mi} \vee AC_{bi}, AC_{mi,bi} \rightarrow \overline{AC_b}, AC_{mi,bi} \rightarrow \overline{AC_m}$$

The first clause says that at least one ORD-Horn partition must hold; the others relate each ORD-Horn partition to its component base variables. Note that we are only requiring that one or more of these base variables be *true*. This effectively means that all refinements of each ORD-Horn relation can potentially be generated.

For edges AB and BC, the base variables double as branch variables; hence we have the usual "at-least-one" and "at-most-one" clauses, shown below for edge AB:

$$AB_m \vee AB_s \vee AB_f \vee AB_{bi}, \overline{AB_m} \vee \overline{AB_s}, \overline{AB_m} \vee \overline{AB_f}$$
$$\overline{AB_m} \vee \overline{AB_{bi}}, \overline{AB_s} \vee \overline{AB_f}, \overline{AB_s} \vee \overline{AB_{bi}}, \overline{AB_f} \vee \overline{AB_{bi}}$$

In general, for each edge, we have one clause saying that at least one of the ORD-Horn partitions holds; and for each of these partitions, we have clauses saying that it implies that one or more of its component base variables are *true* and that all other base variables are *false*. In case an ORD-Horn partition is atomic, the corresponding base variable is used instead of the branch variable (which is not created). Henceforth, when we refer to branch variables, we assume that base variables have been substituted where applicable.

The second group of clauses encode the path consistency of the refinement Φ'' of the network Φ induced by the instantiation of the base variables, by encoding every triple of nodes according to Definition 1. Continuing with Figure 2, we need to encode the condition $\ell_\Phi(i,k) \subseteq \ell_\Phi(i,j) \circ \ell_\Phi(j,k)$ in three orientations: ABC, BCA, CAB (the other three are redundant). The clauses for the first orientation consist of the following (plus an analogous set for AC_{mi} and AC_{bi}):

$$AC_b \rightarrow a_1 \vee a_2 \vee a_3 \vee a_4 \vee a_5 \vee a_6$$
$$AC_m \rightarrow a_7 \vee a_3 \vee a_6, a_1 \rightarrow AB_m, a_1 \rightarrow BC_{fi}$$
$$a_2 \rightarrow AB_m, a_2 \rightarrow BC_b, a_3 \rightarrow AB_s, a_3 \rightarrow BC_{fi}, a_4 \rightarrow AB_s, a_4 \rightarrow BC_b$$
$$a_5 \rightarrow AB_f, a_5 \rightarrow BC_b, a_6 \rightarrow AB_{bi}, a_6 \rightarrow BC_b, a_7 \rightarrow AB_m, a_7 \rightarrow BC_{si}$$

Note that we have utilized a set of auxiliary variables a_i so that the CNF will stay compact. In general, each a_i "defines" a pair of base relations from edge (i,j) and edge (j,k) respectively whose composition contains the encoded base relation from edge (i,k), which is forced to imply the disjunction of the complete list of such pairs. This ensures that the condition $\ell_\Phi(i,k) \subseteq \ell_\Phi(i,j) \circ \ell_\Phi(j,k)$ is met no matter how many base variables are set to *true* for edge (i,k), given that composition distributes over union.

The first group of clauses for every edge and the second for every node triple make up our encoding of an IA network Φ, which we shall refer to as CNF(Φ). Clearly, CNF(Φ) has polynomial size. We proceed next to establish its correctness, that CNF(Φ) is satisfiable iff the network Φ is consistent.

3.3 Correctness of CNF(Φ)

First, if the IA network Φ is consistent, then it has a path-consistent ORD-Horn refinement Φ''. The following lemma implies that every ORD-Horn refinement of (the relation on) an edge must be contained in one of the maximal ORD-Horn partitions of that edge and hence must correspond to a unique branch variable in CNF(Φ):

Lemma 1 *If p and q are nonintersecting ORD-Horn relations such that $p \cup q \notin$ ORD-Horn, $\emptyset \neq p' \subseteq p$, and $\emptyset \neq q' \subseteq q$, then $p' \cup q' \notin$ ORD-Horn.*

One may thus verify that CNF(Φ) is satisfied by setting all the base and branch variables precisely in accordance with Φ'', and then setting to *true* all the auxiliary variables whose values are not forced by unit propagation.

The other direction requires more thinking. If CNF(Φ) is satisfiable, let π be a satisfying assignment. The instantiation of the branch variables in π corresponds to an ORD-Horn refinement Φ' of the IA network Φ, and the instantiation of the base variables in π corresponds to a refinement Φ'' of Φ'. The clauses of CNF(Φ) ensure that Φ'' is path consistent. However, none of the clauses say that Φ'' must be an ORD-Horn network! So how do we know if Φ'' is consistent? And without that knowledge, how do we know that Φ is consistent?

To arrive at the desired conclusion, we need to make a few observations about CNF(Φ). To facilitate the statement of these, if α is a partial or complete instantiation of the branch variables of CNF(Φ), we will write $\Phi|_\alpha$ to denote the corresponding (ORD-Horn) refinement of the IA network Φ noting that if α sets multiple branch variables of an edge to *true*, then $\Phi|_\alpha$ is defined to have the empty relation on every edge.

By examining the clauses of CNF(Φ), let us observe first that enforcement of path consistency on Φ is fully simulated by unit propagation on CNF(Φ). More precisely, we have

Lemma 2 *For any partial or complete instantiation α of the branch variables of* CNF(Φ)*, unit propagation on* CNF(Φ)$|_\alpha$ (i) *produces a contradiction iff enforcing path consistency on $\Phi|_\alpha$ generates an empty relation, and* (ii) *sets a base variable AB_p to false iff enforcing path consistency on $\Phi|_\alpha$ removes the corresponding base relation p from the corresponding edge AB in Φ.*

Our second observation concerns an interesting syntactic property of CNF(Φ) relating to renamable-Horn formulas, which are CNF formulas that can be made to have at most one positive literal per clause by swapping variables with their negations. Specifically, the following may be easily verified (recall that branch variables include base variables where the ORD-Horn partitions are atomic):

Lemma 3 CNF(Φ)$|_\alpha$ *is renamable-Horn for any complete instantiation α of the branch variables of* CNF(Φ).

It is well known that renamable-Horn formulas are satisfiable iff unit propagation does not produce a contradiction. Hence by Lemmas 2 and 3, we have

Lemma 4 *For any complete instantiation α of the branch variables of* CNF(Φ)*,* CNF(Φ)$|_\alpha$ *is satisfiable iff $\Phi|_\alpha$ is consistent.*

We are now well-equipped to complete our argument for the correctness of CNF(Φ). Picking up where we left off, let α be the part of the satisfying assignment π that instantiates the branch variables.

The existence of π implies that CNF$(\Phi)|_\alpha$ is satisfiable, which implies, by Lemma 4, that $\Phi|_\alpha$, and hence Φ, are consistent. Thus we have circumvented the need to reason about the consistency of Φ'', and can now state formally

Theorem 5 CNF(Φ) *is satisfiable iff the network* Φ *is consistent.*

4 Simulation of Native Search by SAT

We will now show that native search (Algorithm 1) on an IA network Φ can be simulated by DPLL on CNF(Φ). To that end it is important to establish that the branching of Algorithm 1 can be simulated by the branching of DPLL (on branch variables).

By Lemma 1, as discussed earlier, any ORD-Horn partition r chosen by Algorithm 1 (line 9) must be contained in an original ORD-Horn partition q, and hence must correspond to a branch variable Q in CNF(Φ).

By Lemma 2, any base relations that may have been removed from the original ORD-Horn partition q by path consistency correspond to base variables that have been set to *false* in CNF(Φ). Hence the choice of r by Algorithm 1 is precisely simulated by setting the branch variable Q to *true* in CNF(Φ). Moreover, by Lemma 2, any branch of Algorithm 1 will terminate precisely when the corresponding branch of DPLL terminates. Hence we have

Theorem 6 *If Algorithm 1 refutes an IA network* Φ *with a search tree of size* n*, then DPLL can refute* CNF(Φ) *with a search tree of size* $O(n)$*.*

Note that the search tree size measures the number of search nodes, while unit propagation occurs within each search node.The constant factor implicit in $O(n)$ in the theorem reflects the fact that DPLL has to simulate a k-way branching of the native search with a sequence of $(k-1)$ binary branchings.

5 Pebbling Contradictions PC$_n$ and Their Reduction to IA Networks

We have now established that SAT is at least as powerful as native search for deciding the consistency of IA networks. The next order of business is constructing a class of IA networks to show that the reverse does not hold. This will be done by enlisting a class of CNF formulas PC$_n$, known as *pebbling contradictions* [4], and then reducing them to IA networks.

For space constraints we will not reproduce the definition of PC$_n$ here, but only note that it is an infinite class of 4CNF formulas (clauses of size < 4 can be "padded" into size 4) that have been shown to admit polynomial-size resolution refutations, but only exponential-size tree-like resolution refutations. In other words, they are tractable for clause learning, but intractable for DPLL.

It remains to translate the 4CNF formulas PC$_n$ into IA networks without creating opportunities for short tree-style refutations, hence producing a class of IA networks that cannot be refuted by native search in polynomial time. Our translation is a general 4CNF-to-IA reduction, based on modifying and extending a known 3CNF-to-IA reduction [18]. We now describe it in detail.

Given a 4CNF formula Δ, our IA network, denoted Φ_Δ, contains exactly the following intervals (variables): a single interval $\mathbf{mid}$; two *literal intervals* $\mathbf{A}$, $\mathbf{notA}$, plus an auxiliary interval $\mathbf{AXnotA}$, for each variable A in Δ; four *literal wrappers* $\mathbf{forP}$, $\mathbf{forQ}$, $\mathbf{forR}$, $\mathbf{forS}$, plus an auxiliary interval $\mathbf{PQRS}$, for each clause $P \vee Q \vee R \vee S$ in Δ, where P, Q, R, and S are (positive or negative) literals.

We describe next the constraints of Φ_Δ, where we may intuitively think of intervals falling before $\mathbf{mid}$ as *false* and those falling after it as *true*. First, for each pair of $\mathbf{A}$ and $\mathbf{notA}$, we ensure that exactly one of them is *true* and the other is *false*, using the third interval $\mathbf{AXnotA}$ ("X" for "excludes") as an auxiliary:

$$\mathbf{mid}\ \{d\}\ \mathbf{AXnotA} \tag{1}$$
$$\mathbf{A}\ \{m, mi\}\ \mathbf{AXnotA} \tag{2}$$
$$\mathbf{notA}\ \{m, mi\}\ \mathbf{AXnotA} \tag{3}$$
$$\mathbf{A}\ \{b, bi\}\ \mathbf{notA} \tag{4}$$

Now for each clause $P \vee Q \vee R \vee S$, the goal is to ensure that at least one of the literals is *true*, i.e., falls after $\mathbf{mid}$. For the same reason for which we have "wrapped" $\mathbf{mid}$ in $\mathbf{AXnotA}$ above, we do not directly constrain the literal intervals $\mathbf{P}, \mathbf{Q}, \mathbf{R}, \mathbf{S}$, but throw them in their respective wrappers which are then to be constrained:

$$\mathbf{P}\ \{d\}\ \mathbf{forP} \tag{5}$$
$$\mathbf{Q}\ \{d\}\ \mathbf{forQ}, \mathbf{R}\ \{d\}\ \mathbf{forR}, \mathbf{S}\ \{d\}\ \mathbf{forS}$$

The following ensures that the wrappers themselves also fall either entirely before (b, m) or after (bi) $\mathbf{mid}$:

$$\mathbf{forP}\ \{b, m, bi\}\ \mathbf{mid} \tag{6}$$
$$\mathbf{forQ}\ \{b, m, bi\}\ \mathbf{mid}, \mathbf{forR}\ \{b, m, bi\}\ \mathbf{mid}, \mathbf{forS}\ \{b, m, bi\}\ \mathbf{mid}$$

And the wrappers for each clause must not overlap:

$$\mathbf{forP}\ \{b, m, mi, bi\}\ \mathbf{forQ} \tag{7}$$
$$\mathbf{forP}\ \{b, m, mi, bi\}\ \mathbf{forR}$$
$$\mathbf{forP}\ \{b, m, mi, bi\}\ \mathbf{forS}, \mathbf{forQ}\ \{b, m, mi, bi\}\ \mathbf{forR}$$
$$\mathbf{forQ}\ \{b, m, mi, bi\}\ \mathbf{forS}, \mathbf{forR}\ \{b, m, mi, bi\}\ \mathbf{forS}$$

The final group of constraints are the essential ones ensuring that at least one of the four literal wrappers (and hence literals) falls after $\mathbf{mid}$. This is achieved through the auxiliary interval $\mathbf{PQRS}$, placed before and adjacent to $\mathbf{mid}$:

$$\mathbf{PQRS}\ \{m\}\ \mathbf{mid} \tag{8}$$

The wrappers having been required not to overlap, the remaining four constraints, together with the above, will ensure that at most three of them can fit before $\mathbf{mid}$:

$$\mathbf{forP}\ \{m, s, f, bi\}\ \mathbf{PQRS} \tag{9}$$
$$\mathbf{forQ}\ \{m, s, f, bi\}\ \mathbf{PQRS} \tag{10}$$
$$\mathbf{forR}\ \{m, s, f, bi\}\ \mathbf{PQRS}, \mathbf{forS}\ \{m, s, f, bi\}\ \mathbf{PQRS}$$

It is straightforward to verify that the reduction described above correctly preserves satisfiability:

Theorem 7 *The IA network* Φ_Δ *is consistent iff the 4CNF formula* Δ *is satisfiable.*

It is important to also establish that if the 4CNF formulas are hard for tree-style search, the resulting IA networks will remain so. To that end, given an unsatisfiable 4CNF formula Δ and its reduction to an IA network Φ_Δ, let us examine the possible decisions (line 9) that can be made by Algorithm 1 running on Φ_Δ. The relations used in Φ_Δ, except the singletons and implicit universal relation, are all non-ORD-Horn. For each of these the partitioning into ORD-Horn relations is as follows:

$$\begin{array}{rcl} \{m, mi\} & \longrightarrow & \{m\}, \{mi\} \\ \{b, bi\} & \longrightarrow & \{b\}, \{bi\} \\ \{b, m, bi\} & \longrightarrow & \{b, m\}, \{bi\} \\ \{m, s, f, bi\} & \longrightarrow & \{m\}, \{s\}, \{f\}, \{bi\} \\ \{b, m, mi, bi\} & \longrightarrow & \{b, m\}, \{mi, bi\} \end{array}$$

We shall argue that in each of the above cases a branch taken by Algorithm 1 (line 9) on Φ_Δ is simulated by a corresponding branch of DPLL on Δ. The first three cases are straightforward: The three partitionings all represent a decision regarding the placement of opposite literal intervals ($\mathbf{A}$, $\mathbf{notA}$) relative to $\mathbf{mid}$, which corresponds to choosing a truth value for a Boolean variable (A). The

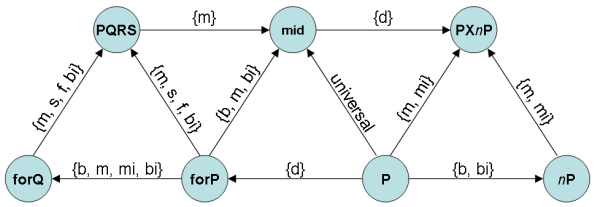

Figure 3. A subgraph of Φ_Δ.

fourth corresponds to choosing a truth value for a Boolean literal, choosing *true* if the branch $\{bi\}$ is taken, and *false* if any of the other three branches $\{m\}, \{s\}, \{f\}$ is taken. The final partitioning concerns the linear ordering of the four literal intervals for a clause, and is the only one that does not directly correspond to decisions on a Boolean variable in Δ. It is safe to pretend, however, that Algorithm 1 never makes decisions with respect to this partitioning, because any such decisions would only be relevant insofar as they cause one of the four intervals to come last and hence after $\mathbf{mid}$, in which case they can simply be replaced by a direct decision placing that interval after $\mathbf{mid}$. To sum up, we have

Lemma 8 *Given an unsatisfiable 4CNF formula Δ and its reduction to an IA network Φ_Δ, if Algorithm 1 refutes Φ_Δ with a search tree of size n, then DPLL can refute Δ with a search tree of size $\leq n$.*

Given that the shortest DPLL refutations of PC_n formulas have exponential size, it follows from Lemma 8 that the IA networks Φ_{PC_n} are intractable for native search:

Theorem 9 *Refutations of Φ_{PC_n} networks by Algorithm 1 take at least exponential time.*

6 Tractability of Φ_{PC_n} for SAT

By now we have constructed a class of IA networks, Φ_{PC_n}, that are provably hard for native search. It remains to show that they are tractable for SAT in that our SAT encoding of them admits polynomial-size resolution refutations and hence can be refuted by clause learning in polynomial time.

Given any 4CNF formula Δ and its reduction to an IA network Φ_Δ, our SAT encoding $\mathrm{CNF}(\Phi_\Delta)$ of Φ_Δ will look rather different from Δ. However, we will show that we can produce in polynomial time, by resolution on $\mathrm{CNF}(\Phi_\Delta)$, a 4CNF formula that is isomorphic to Δ. This will imply that $\mathrm{CNF}(\Phi_\Delta)$ is tractable for resolution and hence clause learning if Δ is.

Let $P \vee Q \vee R \vee S$ be any clause of Δ. For the corresponding intervals $\mathbf{P}, \mathbf{Q}, \mathbf{R}, \mathbf{S}$ of Φ_Δ, we write $n\mathbf{P}, n\mathbf{Q}, n\mathbf{R}, n\mathbf{S}$ for the "opposite" intervals. For example, if P is positive, then $n\mathbf{P} = \mathbf{not}\mathbf{P}$; if Q is negative and hence $\mathbf{Q}$ is some $\mathbf{not}\mathbf{A}$, then $n\mathbf{Q} = \mathbf{A}$.

Instantiating Constraints 1–4 for $\mathbf{P}$, we have

$$\mathbf{mid}\ \{d\}\ \mathbf{PX}n\mathbf{P} \tag{11}$$
$$\mathbf{P}\ \{m, mi\}\ \mathbf{PX}n\mathbf{P} \tag{12}$$
$$n\mathbf{P}\ \{m, mi\}\ \mathbf{PX}n\mathbf{P} \tag{13}$$
$$\mathbf{P}\ \{b, bi\}\ n\mathbf{P} \tag{14}$$

We now proceed to describe elements of $\mathrm{CNF}(\Phi_\Delta)$ that will allow us to derive by resolution a formula isomorphic to Δ. These are the path-consistency-encoding clauses generated for the five types of triangles of Φ_Δ illustrated in Figure 3 (unit propagation has been applied to simplify some of the clauses).

6.1 Elements of $\mathrm{CNF}(\Phi_\Delta)$

First consider the triangle $\langle \mathbf{P}, n\mathbf{P}, \mathbf{PX}n\mathbf{P}\rangle$ in Φ_Δ, labeled with Constraints 12–14. Let $P_\bot$ and $P_\top$ be the base variables of $\mathrm{CNF}(\Phi_\Delta)$ encoding the edge $(\mathbf{P}, n\mathbf{P})$, and P_mX and $P_{mi}X$ those encoding the edge $(\mathbf{P}, \mathbf{PX}n\mathbf{P})$. The clauses encoding this triangle include

$$P_mX \to P_\bot$$
$$P_{mi}X \to P_\top \tag{15}$$
$$P_\bot \vee P_\top$$
$$\overline{P_\bot} \vee \overline{P_\top} \tag{16}$$

Consider next the triangle $\langle \mathbf{P}, \mathbf{mid}, \mathbf{PX}n\mathbf{P}\rangle$. The edge $(\mathbf{mid}, \mathbf{PX}n\mathbf{P})$ is labeled with Constraint 11. The constraint between $\mathbf{P}$ and $\mathbf{mid}$ is universal, and hence will generate 13 Boolean variables, among which are PM_b and PM_{bi}. The clauses encoding this triangle include

$$PM_b \to P_mX$$
$$PM_{bi} \to P_{mi}X \tag{17}$$

Now consider the triangle $\langle \mathbf{forP}, \mathbf{forQ}, \mathbf{PQRS}\rangle$, labeled with Constraints 7, 9, and 10. Let PQ_b, PQ_m, PQ_{mi}, and PQ_{bi} be the base variables for the edge $(\mathbf{forP}, \mathbf{forQ})$; P_m, P_s, P_f, and P_{bi} those for $(\mathbf{forP}, \mathbf{PQRS})$; and Q_m, Q_s, Q_f, and Q_{bi} those for $(\mathbf{forQ}, \mathbf{PQRS})$. Our SAT encoding of this triangle contains three groups of clauses, one for each orientation of path consistency. Only two of them are needed here. These are

$$P_m \to b_1 \vee b_2 \vee b_3 \tag{18}$$
$$P_s \to b_1 \vee b_2 \vee b_4 \vee b_5, P_f \to b_2 \vee b_6 \vee b_7 \vee b_8, P_{bi} \to b_7 \vee b_8 \vee b_9 \vee b_{10}$$
$$b_1 \to PQ_b, b_1 \to Q_f \tag{19}$$
$$b_2 \to PQ_b, b_2 \to Q_{bi} \tag{20}$$
$$b_3 \to PQ_m, b_3 \to Q_s \tag{21}$$
$$b_4 \to PQ_m, b_4 \to Q_f, b_5 \to PQ_{mi}, b_5 \to Q_m, b_6 \to PQ_{mi}, b_6 \to Q_s$$
$$b_7 \to PQ_{bi}, b_7 \to Q_m, b_8 \to PQ_{bi}, b_8 \to Q_s, b_9 \to PQ_{bi}, b_9 \to Q_f$$
$$b_{10} \to PQ_{bi}, b_{10} \to Q_{bi}$$
$$P_m \vee P_s \vee P_f \vee P_{bi} \tag{22}$$
$$\overline{P_m} \vee \overline{P_s}, \overline{P_m} \vee \overline{P_f}, \overline{P_m} \vee \overline{P_{bi}} \tag{23}$$
$$\overline{P_s} \vee \overline{P_f}, \overline{P_s} \vee \overline{P_{bi}}, \overline{P_f} \vee \overline{P_{bi}}$$

and another group analogous to the above, obtained by swapping $\{P_m, P_s, P_f, P_{bi}\}$ respectively with $\{Q_m, Q_s, Q_f, Q_{bi}\}$ and renaming the auxiliary variables b_i. In particular, the second group contains the following clauses analogous to Clauses 23:

$$\overline{Q_m} \vee \overline{Q_s}, \overline{Q_m} \vee \overline{Q_f}, \overline{Q_m} \vee \overline{Q_{bi}} \tag{24}$$

Referring to Clause 22 and writing down the analogous clauses for $\{Q, R, S\}$, we have also the following in $\mathrm{CNF}(\Phi_\Delta)$:

$$Q_m \vee Q_s \vee Q_f \vee Q_{bi} \tag{25}$$
$$R_m \vee R_s \vee R_f \vee R_{bi} \tag{26}$$
$$S_m \vee S_s \vee S_f \vee S_{bi} \tag{27}$$

Consider next the triangle $\langle \mathbf{forP}, \mathbf{PQRS}, \mathbf{mid}\rangle$. Constraint 8, between $\mathbf{PQRS}$ and $\mathbf{mid}$, is a singleton and need not be encoded. Constraint 6, between $\mathbf{forP}$ and $\mathbf{mid}$, generates three base variables fPM_b, fPM_m, fPM_{bi}. The clauses encoding this triangle include

$$P_m \to fPM_b, P_s \to fPM_b, P_f \to fPM_m$$
$$P_{bi} \to fPM_{bi} \tag{28}$$

Finally, consider the triangle $\langle \mathbf{P}, \mathbf{forP}, \mathbf{mid}\rangle$. Constraint 5, between $\mathbf{P}$ and $\mathbf{forP}$, is a singleton and hence need not be encoded. As discussed earlier, the constraint between $\mathbf{P}$ and $\mathbf{mid}$ is universal, generating 13 Boolean variables including PM_b and PM_{bi}. The clauses encoding this triangle include

$$fPM_b \to PM_b, fPM_m \to PM_b$$
$$fPM_{bi} \to PM_{bi} \tag{29}$$

6.2 A Resolution Derivation from $\mathrm{CNF}(\Phi_\Delta)$

From Clauses 18, 19, 20, 21, and 24 one may derive: $\overline{P_m} \vee \overline{Q_m}$. By symmetry, the following may all be derived from $\mathrm{CNF}(\Phi_\Delta)$:

$$\overline{P_s} \vee \overline{Q_s}, \overline{P_f} \vee \overline{Q_f}, \overline{P_m} \vee \overline{R_m}, \overline{P_s} \vee \overline{R_s}, \overline{P_f} \vee \overline{R_f}$$
$$\overline{P_m} \vee \overline{S_m}, \overline{P_s} \vee \overline{S_s}, \overline{P_f} \vee \overline{S_f}, \overline{Q_m} \vee \overline{R_m}, \overline{Q_s} \vee \overline{R_s}, \overline{Q_f} \vee \overline{R_f}$$
$$\overline{Q_m} \vee \overline{S_m}, \overline{Q_s} \vee \overline{S_s}, \overline{Q_f} \vee \overline{S_f}, \overline{R_m} \vee \overline{S_m}, \overline{R_s} \vee \overline{S_s}, \overline{R_f} \vee \overline{S_f}$$

The following derivation can now be readily produced:

($\overline{P_m} \vee \overline{Q_m}$ with 22)	$\overline{Q_m} \vee P_s \vee P_f \vee P_{bi}$	
(25 with above)	$Q_s \vee Q_f \vee Q_{bi} \vee P_s \vee P_f \vee P_{bi}$	
($\overline{Q_s} \vee \overline{R_s}$ with above)	$\overline{R_s} \vee Q_f \vee Q_{bi} \vee P_s \vee P_f \vee P_{bi}$	
($\overline{P_s} \vee \overline{R_s}$ with above)	$\overline{R_s} \vee Q_f \vee Q_{bi} \vee P_f \vee P_{bi}$	
($\overline{Q_f} \vee \overline{S_f}$ with above)	$\overline{R_s} \vee \overline{S_f} \vee Q_{bi} \vee P_f \vee P_{bi}$	
($\overline{P_f} \vee \overline{S_f}$ with above)	$\overline{R_s} \vee \overline{S_f} \vee Q_{bi} \vee P_{bi}$	(30)
($\overline{R_f} \vee \overline{S_f}$ with 26)	$R_m \vee R_s \vee \overline{S_f} \vee R_{bi}$	
(30 with above)	$R_m \vee \overline{S_f} \vee Q_{bi} \vee P_{bi} \vee R_{bi}$	(31)
($\overline{P_m} \vee \overline{R_m}$ with 22)	$\overline{R_m} \vee P_s \vee P_f \vee P_{bi}$	
($\overline{P_f} \vee \overline{S_f}$ with above)	$\overline{R_m} \vee P_s \vee \overline{S_f} \vee P_{bi}$	(32)
($\overline{Q_m} \vee \overline{R_m}$ with 25)	$\overline{R_m} \vee Q_s \vee Q_f \vee Q_{bi}$	
($\overline{Q_f} \vee \overline{S_f}$ with above)	$\overline{R_m} \vee Q_s \vee \overline{S_f} \vee Q_{bi}$	
($\overline{P_s} \vee \overline{Q_s}$ with above)	$\overline{R_m} \vee \overline{P_s} \vee \overline{S_f} \vee Q_{bi}$	
(32 with above)	$\overline{R_m} \vee \overline{S_f} \vee P_{bi} \vee Q_{bi}$	
(31 with above)	$\overline{S_f} \vee P_{bi} \vee Q_{bi} \vee R_{bi}$	(33)

By analogy with Clause 33, the following may also be derived:

$$\overline{S_s} \vee P_{bi} \vee Q_{bi} \vee R_{bi} \tag{34}$$
$$\overline{S_m} \vee P_{bi} \vee Q_{bi} \vee R_{bi} \tag{35}$$

Resolving Clauses 27, 33, 34, and 35 gives $P_{bi} \vee Q_{bi} \vee R_{bi} \vee S_{bi}$. Resolving this with Clauses 28 and 29 and the analogous clauses for $\{Q, R, S\}$ gives $PM_{bi} \vee QM_{bi} \vee RM_{bi} \vee SM_{bi}$. Resolving this with Clauses 17 and 15 and the analogous clauses for $\{Q, R, S\}$ gives

$$P_\top \vee Q_\top \vee R_\top \vee S_\top \tag{36}$$

In contrast to the original clause $P \vee Q \vee R \vee S$ of Δ, Clause 36 only has positive literals. However, resolving it with Clause 16 and the analogous clauses for $\{Q, R, S\}$ as necessary, we can turn it into a clause that is isomorphic to $P \vee Q \vee R \vee S$. This final clause may be formally given as

$$P' \vee Q' \vee R' \vee S' \tag{37}$$

where P' is $P_\top$ if P is positive and $\overline{P_\bot}$ otherwise, and similarly for Q', R', and S'.

It is thus clear that repeating the derivation of Clause 37 for each clause $P \vee Q \vee R \vee S$ in Δ will produce a 4CNF formula isomorphic to Δ. Each repetition takes a constant number of steps, and hence the whole process clearly takes only polynomial time. Since the PC_n formulas are known to have polynomial-size resolution refutations, substituting PC_n for Δ, it follows that our SAT encoding of the Φ_{PC_n} networks also have polynomial-size resolution refutations. Since clause learning is as powerful as resolution, we have finally arrived at the end of our investigation:

Theorem 10 $\mathrm{CNF}(\Phi_{PC_n})$ *can be refuted by clause learning in polynomial time.*[4]

7 Conclusion and Future Work

Theorem 6 establishes that SAT is at least as powerful as native search for deciding the consistency of IA networks. Theorems 9 and 10 together establish that the reverse does not hold: There are IA networks that require exponentially longer refutations by native search than by SAT. Therefore, we conclude that SAT strictly dominates native search in theoretical power for deciding the consistency of IA networks in qualitative temporal reasoning.

While the IA networks Φ_{PC_n} constructed from the pebbling contradictions witness the separation in power of the two approaches to qualitative temporal reasoning, it remains an interesting question whether a similar separation can be associated with any class of random networks as used in standard benchmarking of qualitative reasoners. Another open question is how exactly the use of ORD-Horn (over the set of base relations) as a tractable subset affects the power of the proof system, in both the native and SAT-based approaches.

[4] The reader is reminded that *clause learning* here refers to a proof system—a concrete clause learning solver having a deterministic resolution strategy is not guaranteed to run in polynomial time on these formulas.

REFERENCES

[1] James F. Allen, 'Maintaining knowledge about temporal intervals', *Communications of the ACM*, **26**(11), 832–843, (1983).

[2] Fahiem Bacchus, 'GAC via unit propagation', in *Proceedings of the 13th International Conference on Principles and Practice of Constraint Programming (CP)*, pp. 133–147, (2007).

[3] Paul Beame, Henry A. Kautz, and Ashish Sabharwal, 'Towards understanding and harnessing the potential of clause learning', *Journal of Artificial Intelligence Research*, **22**, 319–351, (2004).

[4] Eli Ben-Sasson, Russell Impagliazzo, and Avi Wigderson, 'Near optimal separation of tree-like and general resolution', *Combinatorica*, **24**(4), 585–603, (2004).

[5] Jean-François Condotta, Dominique DAlmeida, Christophe Lecoutre, and Lakhdar Saïs, 'From qualitative to discrete constraint networks', in *Workshop on Qualitative Constraint Calculi*, pp. 54–64, (2006).

[6] Jean-François Condotta, Gérard Ligozat, and Mahmoud Saade, 'Eligible and frozen constraints for solving temporal qualitative constraint networks', in *Proceedings of the 13th International Conference on Principles and Practice of Constraint Programming (CP)*, pp. 806–814, (2007).

[7] Martin Davis, George Logemann, and Donald Loveland, 'A machine program for theorem proving', *Journal of the ACM*, **(5)7**, 394–397, (1962).

[8] Zeno Gantner, Matthias Westphal, and Stefan Wölfl, 'GQR—A fast reasoner for binary qualitative constraint calculi', in *AAAI-08 Workshop on Spatial and Temporal Reasoning*, (2008).

[9] Jinbo Huang, 'The effect of restarts on the efficiency of clause learning', in *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2318–2323, (2007).

[10] Jason Jingshi Li, Jinbo Huang, and Jochen Renz, 'A divide-and-conquer approach for solving interval algebra networks', in *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 572–577, (2009).

[11] Alan K. Mackworth, 'Consistency in networks of relations', *Artificial Intelligence*, **8**, 99–118, (1977).

[12] Joao Marques-Silva and Karem Sakallah, 'GRASP—A new search algorithm for satisfiability', in *Proceedings of the International Conference on Computer Aided Design (ICCAD)*, pp. 220–227, (1996).

[13] Matthew Moskewicz, Conor Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik, 'Chaff: Engineering an efficient SAT solver', in *Proceedings of the 38th Design Automation Conference (DAC)*, pp. 530–535, (2001).

[14] Bernhard Nebel, 'Solving hard qualitative temporal reasoning problems: Evaluating the efficiency of using the ORD-Horn class', *Constraints*, **1**(3), 175–190, (1997).

[15] Bernhard Nebel and Hans-Jürgen Bürckert, 'Reasoning about temporal relations: A maximal tractable subclass of Allen's interval algebra', *Journal of the ACM*, **42**(1), 43–66, (1995).

[16] Duc Nghia Pham, John Thornton, and Abdul Sattar, 'Modelling and solving temporal reasoning as propositional satisfiability', *Artificial Intelligence*, **172**(15), 1752–1782, (2008).

[17] Knot Pipatsrisawat and Adnan Darwiche, 'On the power of clause-learning SAT solvers with restarts', in *CP*, ed., Ian P. Gent, volume 5732 of *Lecture Notes in Computer Science*, pp. 654–668. Springer, (2009).

[18] Marc Vilain and Henry Kautz, 'Constraint propagation algorithms for temporal reasoning', in *Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI)*, pp. 377–382, (1986).