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

TABLEAUX-2013

Davide Bresolin, Dario Della Monica, Angelo Montanari and
Guido Sciavicco

University of Verona - Italy, Reykjavik University - Iceland
University of Udine - Italy, University of Murcia - Spain

Presentation

- This is an implementation and experimental work, no new theoretical results here.

Presentation

- This is an implementation and experimental work, no new theoretical results here.
- Nevertheless, some experimental work, such as this one, are not free of unexpected problems and difficulties.

Presentation

- This is an implementation and experimental work, no new theoretical results here.
- Nevertheless, some experimental work, such as this one, are not free of unexpected problems and difficulties.
- Moreover, this implementation, although particularly simple, is the only one of its kind.

Time and logics

In AI usually time is formalized with languages (logics) that are:

- point-based, or
- interval-based.

Time and logics

In AI usually time is formalized with languages (logics) that are:

- point-based, or
- interval-based.

In **point-based** temporal logics, formulas are interpreted directly over points. In **interval-based** ones, they are interpreted over **intervals** (our case). We are interested only in qualitative relationships between intervals, and truth of a formula over an interval does not come from nor influences the truth of the same formula over sub-intervals.

Time and logics

In AI usually time is formalized with languages (logics) that are:

- point-based, or
- interval-based.

In **point-based** temporal logics, formulas are interpreted directly over points. In **interval-based** ones, they are interpreted over **intervals** (our case). We are interested only in qualitative relationships between intervals, and truth of a formula over an interval does not come from nor influences the truth of the same formula over sub-intervals.

It is well-known that interval-based logics are much more difficult to deal with.

What is an interval?

Definition

Given a linear order $\mathbb{D} = \langle D, < \rangle$:

- an interval in $\mathbb{D}$ is a pair $[d_0, d_1]$ such that $d_0 < d_1$;
- $\mathbb{I}(\mathbb{D})$ is the set of all (strict) intervals on $\mathbb{D}$;
- $\langle \mathbb{D}, \mathbb{I}(\mathbb{D}) \rangle$ is an interval structure.

What is an interval?

Definition

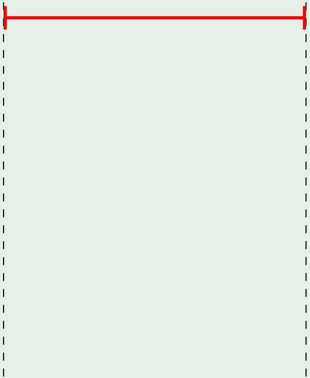
Given a linear order $\mathbb{D} = \langle D, < \rangle$:

- an interval in $\mathbb{D}$ is a pair $[d_0, d_1]$ such that $d_0 < d_1$;
- $\mathbb{I}(\mathbb{D})$ is the set of all (strict) intervals on $\mathbb{D}$;
- $\langle \mathbb{D}, \mathbb{I}(\mathbb{D}) \rangle$ is an interval structure.

- We consider intervals as pairs of time points.
- A point $d \in D$ belongs to $[d_0, d_1]$ if $d_0 \leq d \leq d_1$.
- Sometimes, the non-strict semantics, where intervals may have coincident endpoints, is considered; nowadays, the common choice is to exclude this possibility, treating points as a different sort, and giving rise to more complex point-interval logics.

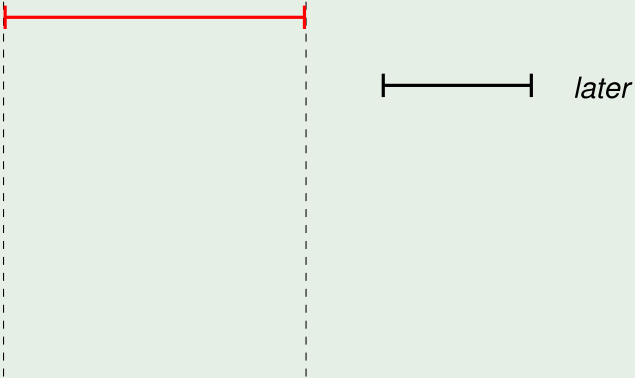
Allen's binary relations

There are 13 different binary relations between intervals:



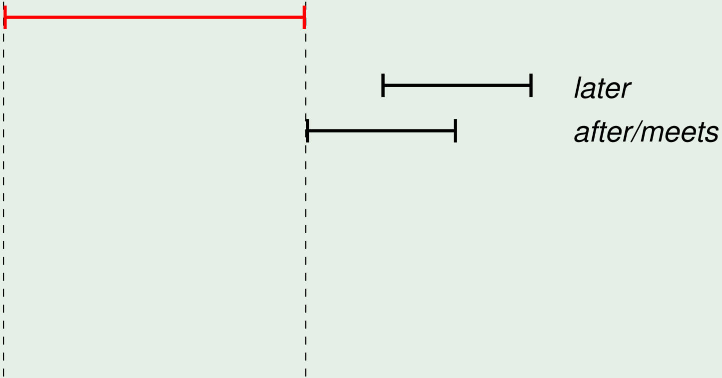
Allen's binary relations

There are 13 different binary relations between intervals:



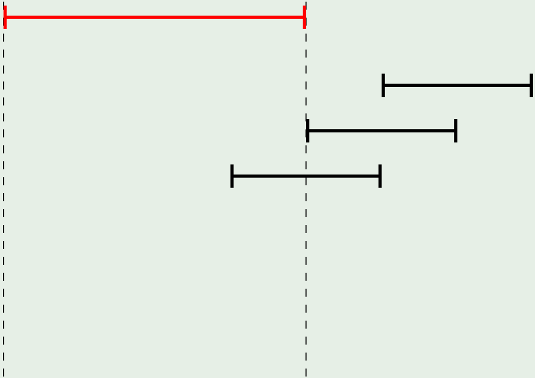
Allen's binary relations

There are 13 different binary relations between intervals:



Allen's binary relations

There are 13 different binary relations between intervals:



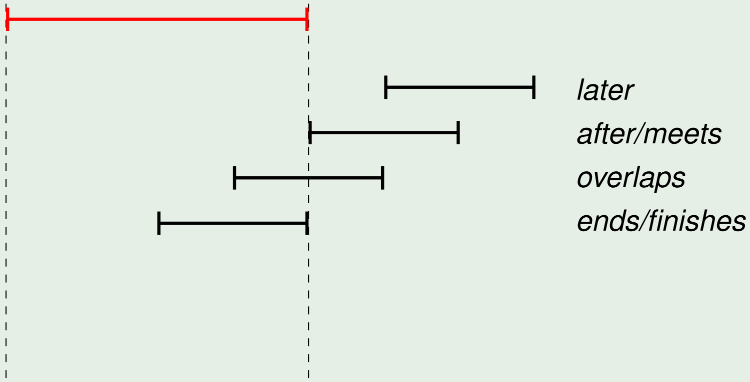
later

after/meets

overlaps

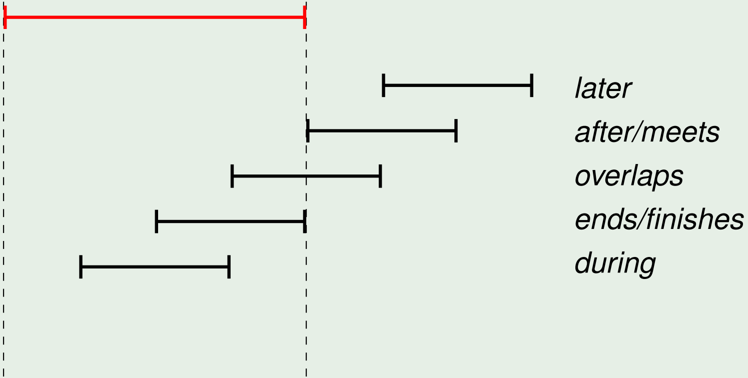
Allen's binary relations

There are 13 different binary relations between intervals:



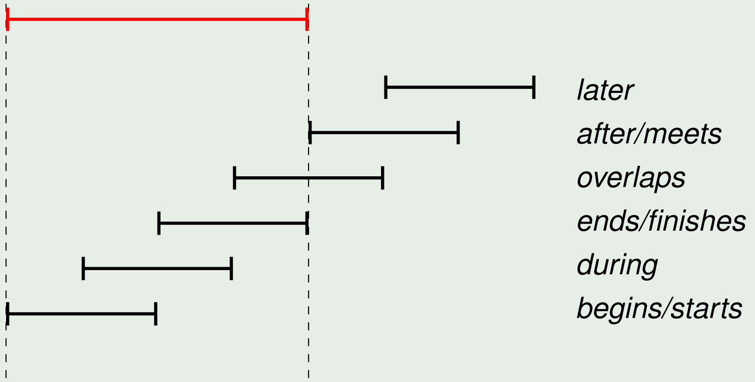
Allen's binary relations

There are 13 different binary relations between intervals:



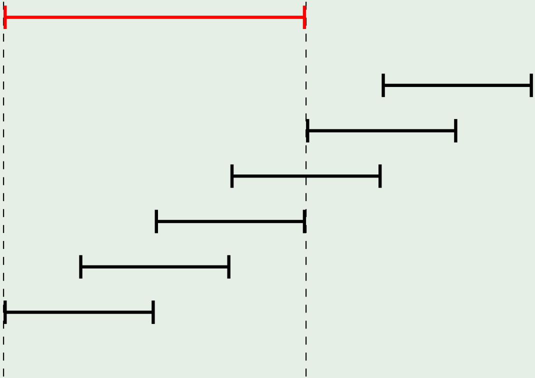
Allen's binary relations

There are 13 different binary relations between intervals:



Allen’s binary relations

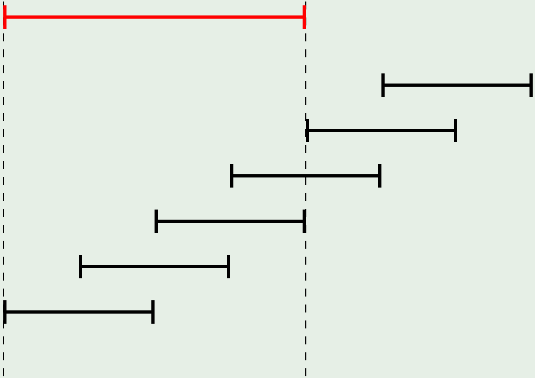
There are 13 different binary relations between intervals:



later	$\langle L \rangle$
after/meets	$\langle A \rangle$
overlaps	$\langle O \rangle$
ends/finishes	$\langle E \rangle$
during	$\langle D \rangle$
begins/starts	$\langle B \rangle$

Allen's binary relations

There are 13 different binary relations between intervals:

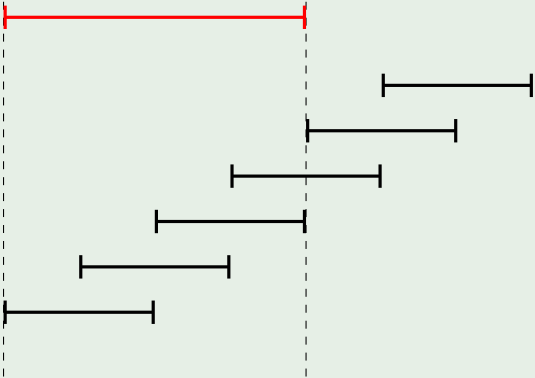


later	$\langle L\rangle$	$\langle \overline{L}\rangle$
after/meets	$\langle A\rangle$	$\langle \overline{A}\rangle$
overlaps	$\langle O\rangle$	$\langle \overline{O}\rangle$
ends/finishes	$\langle E\rangle$	$\langle \overline{E}\rangle$
during	$\langle D\rangle$	$\langle \overline{D}\rangle$
begins/starts	$\langle B\rangle$	$\langle \overline{B}\rangle$

together with their inverses.

Allen's binary relations

There are 13 different binary relations between intervals:



later	$\langle L \rangle$ $\langle \overline{L} \rangle$
after/meets	$\langle A \rangle$ $\langle \overline{A} \rangle$
overlaps	$\langle O \rangle$ $\langle \overline{O} \rangle$
ends/finishes	$\langle E \rangle$ $\langle \overline{E} \rangle$
during	$\langle D \rangle$ $\langle \overline{D} \rangle$
begins/starts	$\langle B \rangle$ $\langle \overline{B} \rangle$

together with their inverses.

Between points we have only three binary ordering relations!

Halpern-Shoham's modal logic of interval relations

Every interval relation gives rise to a modal operator over interval structures.

Halpern-Shoham's modal logic of interval relations

Every interval relation gives rise to a modal operator over interval structures. Thus, a multimodal logic arises:

Syntax of Halpern-Shoham's logic, hereafter called HS :

$$\phi ::= p \mid \neg\phi \mid \phi \wedge \psi \mid \langle B\rangle\phi \mid \langle E\rangle\phi \mid \langle \overline{B}\rangle\phi \mid \langle \overline{E}\rangle\phi \mid \langle A\rangle\phi \mid \langle \overline{A}\rangle\phi.$$

Halpern-Shoham's modal logic of interval relations

Every interval relation gives rise to a modal operator over interval structures. Thus, a multimodal logic arises:

Syntax of Halpern-Shoham's logic, hereafter called HS :

$$\phi ::= p \mid \neg\phi \mid \phi \wedge \psi \mid \langle B\rangle\phi \mid \langle E\rangle\phi \mid \langle \overline{B}\rangle\phi \mid \langle \overline{E}\rangle\phi \mid \langle A\rangle\phi \mid \langle \overline{A}\rangle\phi.$$

Interval model:

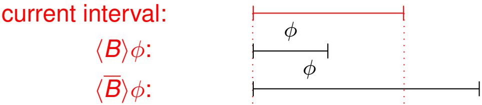
$$\mathbf{M} = \langle \mathbb{I}(\mathbb{D}), V\rangle,$$

where $V : \mathcal{AP} \mapsto 2^{\mathbb{I}(\mathbb{D})}$.

Formal semantics of HS

$\langle B \rangle$: $\mathbf{M}, [d_0, d_1] \Vdash \langle B \rangle \phi$ iff there exists d_2 such that $d_0 \leq d_2 < d_1$ and $\mathbf{M}, [d_0, d_2] \Vdash \phi$.

$\langle \overline{B} \rangle$: $\mathbf{M}, [d_0, d_1] \Vdash \langle \overline{B} \rangle \phi$ iff there exists d_2 such that $d_1 < d_2$ and $\mathbf{M}, [d_0, d_2] \Vdash \phi$.



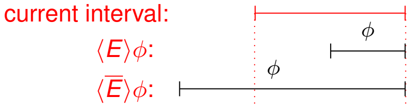
Formal semantics of HS

$\langle B \rangle$: $\mathbf{M}, [d_0, d_1] \Vdash \langle B \rangle \phi$ iff there exists d_2 such that $d_0 \leq d_2 < d_1$ and $\mathbf{M}, [d_0, d_2] \Vdash \phi$.

$\langle \overline{B} \rangle$: $\mathbf{M}, [d_0, d_1] \Vdash \langle \overline{B} \rangle \phi$ iff there exists d_2 such that $d_1 < d_2$ and $\mathbf{M}, [d_0, d_2] \Vdash \phi$.

$\langle E \rangle$: $\mathbf{M}, [d_0, d_1] \Vdash \langle E \rangle \phi$ iff there exists d_2 such that $d_0 < d_2 \leq d_1$ and $\mathbf{M}, [d_2, d_1] \Vdash \phi$.

$\langle \overline{E} \rangle$: $\mathbf{M}, [d_0, d_1] \Vdash \langle \overline{E} \rangle \phi$ iff there exists d_2 such that $d_2 < d_0$ and $\mathbf{M}, [d_2, d_1] \Vdash \phi$.



Formal semantics of HS

$\langle B \rangle$: $\mathbf{M}, [d_0, d_1] \Vdash \langle B \rangle \phi$ iff there exists d_2 such that $d_0 \leq d_2 < d_1$ and $\mathbf{M}, [d_0, d_2] \Vdash \phi$.

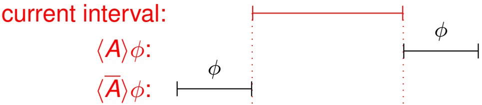
$\langle \overline{B} \rangle$: $\mathbf{M}, [d_0, d_1] \Vdash \langle \overline{B} \rangle \phi$ iff there exists d_2 such that $d_1 < d_2$ and $\mathbf{M}, [d_0, d_2] \Vdash \phi$.

$\langle E \rangle$: $\mathbf{M}, [d_0, d_1] \Vdash \langle E \rangle \phi$ iff there exists d_2 such that $d_0 < d_2 \leq d_1$ and $\mathbf{M}, [d_2, d_1] \Vdash \phi$.

$\langle \overline{E} \rangle$: $\mathbf{M}, [d_0, d_1] \Vdash \langle \overline{E} \rangle \phi$ iff there exists d_2 such that $d_2 < d_0$ and $\mathbf{M}, [d_2, d_1] \Vdash \phi$.

$\langle A \rangle$: $\mathbf{M}, [d_0, d_1] \Vdash \langle A \rangle \phi$ iff there exists d_2 such that $d_1 < d_2$ and $\mathbf{M}, [d_1, d_2] \Vdash \phi$.

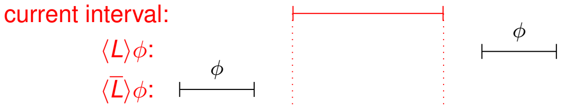
$\langle \overline{A} \rangle$: $\mathbf{M}, [d_0, d_1] \Vdash \langle \overline{A} \rangle \phi$ iff there exists d_2 such that $d_2 < d_0$ and $\mathbf{M}, [d_2, d_0] \Vdash \phi$.



Formal semantics of HS - contd'

$\langle L \rangle$: $\mathbf{M}, [d_0, d_1] \Vdash \langle L \rangle \phi$ iff there exists d_2, d_3 such that $d_1 < d_2 < d_3$ and $\mathbf{M}, [d_2, d_3] \Vdash \phi$.

$\langle \overline{L} \rangle$: $\mathbf{M}, [d_0, d_1] \Vdash \langle \overline{L} \rangle \phi$ iff there exists d_2, d_3 such that $d_2 < d_3 < d_0$ and $\mathbf{M}, [d_2, d_3] \Vdash \phi$.



Formal semantics of HS - contd'

$\langle L \rangle$: $\mathbf{M}, [d_0, d_1] \Vdash \langle L \rangle \phi$ iff there exists d_2, d_3 such that $d_1 < d_2 < d_3$ and $\mathbf{M}, [d_2, d_3] \Vdash \phi$.

$\langle \overline{L} \rangle$: $\mathbf{M}, [d_0, d_1] \Vdash \langle \overline{L} \rangle \phi$ iff there exists d_2, d_3 such that $d_2 < d_3 < d_0$ and $\mathbf{M}, [d_2, d_3] \Vdash \phi$.

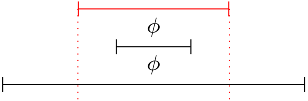
$\langle D \rangle$: $\mathbf{M}, [d_0, d_1] \Vdash \langle D \rangle \phi$ iff there exists d_2, d_3 such that $d_0 < d_2 < d_3 < d_1$ and $\mathbf{M}, [d_2, d_3] \Vdash \phi$.

$\langle \overline{D} \rangle$: $\mathbf{M}, [d_0, d_1] \Vdash \langle \overline{D} \rangle \phi$ iff there exists d_2, d_3 such that $d_2 < d_0 < d_1 < d_3$ and $\mathbf{M}, [d_2, d_3] \Vdash \phi$.

current interval:

$\langle D \rangle \phi$:

$\langle \overline{D} \rangle \phi$:



Formal semantics of HS - contd'

$\langle L \rangle$: $\mathbf{M}, [d_0, d_1] \Vdash \langle L \rangle \phi$ iff there exists d_2, d_3 such that $d_1 < d_2 < d_3$ and $\mathbf{M}, [d_2, d_3] \Vdash \phi$.

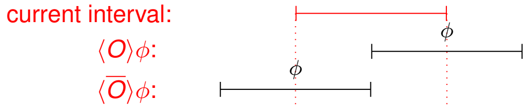
$\langle \overline{L} \rangle$: $\mathbf{M}, [d_0, d_1] \Vdash \langle \overline{L} \rangle \phi$ iff there exists d_2, d_3 such that $d_2 < d_3 < d_0$ and $\mathbf{M}, [d_2, d_3] \Vdash \phi$.

$\langle D \rangle$: $\mathbf{M}, [d_0, d_1] \Vdash \langle D \rangle \phi$ iff there exists d_2, d_3 such that $d_0 < d_2 < d_3 < d_1$ and $\mathbf{M}, [d_2, d_3] \Vdash \phi$.

$\langle \overline{D} \rangle$: $\mathbf{M}, [d_0, d_1] \Vdash \langle \overline{D} \rangle \phi$ iff there exists d_2, d_3 such that $d_2 < d_0 < d_1 < d_3$ and $\mathbf{M}, [d_2, d_3] \Vdash \phi$.

$\langle O \rangle$: $\mathbf{M}, [d_0, d_1] \Vdash \langle O \rangle \phi$ iff there exists d_2, d_3 such that $d_0 < d_2 < d_1 < d_3$ and $\mathbf{M}, [d_2, d_3] \Vdash \phi$.

$\langle \overline{O} \rangle$: $\mathbf{M}, [d_0, d_1] \Vdash \langle \overline{O} \rangle \phi$ iff there exists d_2, d_3 such that $d_2 < d_0 < d_3 < d_1$ and $\mathbf{M}, [d_2, d_3] \Vdash \phi$.



The zoo of fragments of HS

Technically, there are $2^{12} = 4096$ fragments of HS.

The zoo of fragments of HS

Technically, there are $2^{12} = 4096$ fragments of HS.

Even if the syntax presented before displayed only a sub-conjunct of the modalities, one can think as HS as built up using all of them. Some easy (and some very hard) expressivity results allow us to reduce the number of expressively different fragments.

The zoo of fragments of HS

Technically, there are $2^{12} = 4096$ fragments of HS.

Even if the syntax presented before displayed only a sub-conjunct of the modalities, one can think as HS as built up using all of them. Some easy (and some very hard) expressivity results allow us to reduce the number of expressively different fragments.

Expressiveness classification problem: classify the fragments of HS with respect to their expressiveness. (Note: this depends on the properties of the linear order).

The zoo of fragments of HS

Technically, there are $2^{12} = 4096$ fragments of HS.

Even if the syntax presented before displayed only a sub-conjunct of the modalities, one can think as HS as built up using all of them. Some easy (and some very hard) expressivity results allow us to reduce the number of expressively different fragments.

Expressiveness classification problem: classify the fragments of HS with respect to their expressiveness. (Note: this depends on the properties of the linear order).

Decidability problem: classify the fragments of HS with respect to decidability/complexity status of their satisfiability problem. (Note: this depends even more of the properties of the linear order).

The zoo of fragments of HS

Technically, there are $2^{12} = 4096$ fragments of HS.

Even if the syntax presented before displayed only a sub-conjunct of the modalities, one can think as HS as built up using all of them. Some easy (and some very hard) expressivity results allow us to reduce the number of expressively different fragments.

Expressiveness classification problem: classify the fragments of HS with respect to their expressiveness. (Note: this depends on the properties of the linear order).

Decidability problem: classify the fragments of HS with respect to decidability/complexity status of their satisfiability problem. (Note: this depends even more of the properties of the linear order).
We focus here on the class of all finite linearly ordered sets.

Classifying HS fragments on finite linear orders - 1

Complexity class:

1: Non primitive recursive

2: EXPSPACE-complete

3: NEXPTIME-complete

4: NP-complete

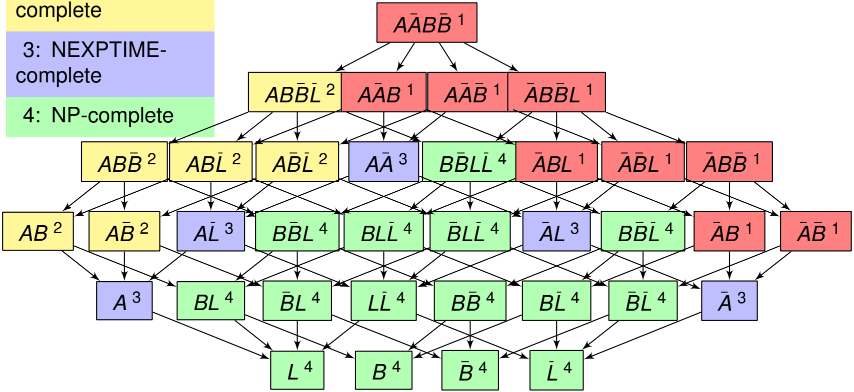


Figure : Hasse diagram of all and only decidable fragments of HS over finite linear orders.

Classifying HS fragments on finite linear orders - 2

In the previous figure, only the decidable cases are displayed, along with their complexity.

Classifying HS fragments on finite linear orders - 2

In the previous figure, only the decidable cases are displayed, along with their complexity.

They are 62, out of 1347 expressively different fragment in the finite case.

Classifying HS fragments on finite linear orders - 2

In the previous figure, only the decidable cases are displayed, along with their complexity.

They are 62, out of 1347 expressively different fragment in the finite case.

F.Y.I., on this web address: *https://itl.dimi.uniud.it/content/logic-hs* you can find any information about HS-fragments and their fragments, updated to the latest advances, which are the results of over 10 years of research in this topic.

Classifying HS fragments on finite linear orders - 2

In the previous figure, only the decidable cases are displayed, along with their complexity.

They are 62, out of 1347 expressively different fragment in the finite case.

F.Y.I., on this web address: *https://itl.dimi.uniud.it/content/logic-hs* you can find any information about HS-fragments and their fragments, updated to the latest advances, which are the results of over 10 years of research in this topic.

In this work, we use the results we have that concern the fragment A in the finite case, and we put them at work to build a usable satisfiability checker.

The theoretical basis for $\mathcal{A}$-formulas satisfiability checking - 1

The first semi-decidability procedure for A has been published in 2003.

The theoretical basis for A-formulas satisfiability checking - 1

The first semi-decidability procedure for A has been published in 2003.

In 2007 it has been proved that A is decidable in the finite and in the strongly discrete case (actually, the result concerned the more expressive fragment $A\overline{A}$).

The theoretical basis for A-formulas satisfiability checking - 1

The first semi-decidability procedure for A has been published in 2003.

In 2007 it has been proved that A is decidable in the finite and in the strongly discrete case (actually, the result concerned the more expressive fragment $A\overline{A}$).

The 2007 result is based on a small-model theorem, and gives rise to a declarative tableaux method. Such a technique is asymptotically optimal, but unusable in practice.

The theoretical basis for A-formulas satisfiability checking - 1

The first semi-decidability procedure for A has been published in 2003.

In 2007 it has been proved that A is decidable in the finite and in the strongly discrete case (actually, the result concerned the more expressive fragment $A\overline{A}$).

The 2007 result is based on a small-model theorem, and gives rise to a declarative tableaux method. Such a technique is asymptotically optimal, but unusable in practice.

In this work we adapted the 2003 result (a classical tableaux) with the closing condition from the 2007 result, to obtain a usable, optimizable, and relatively easy to implement method.

The theoretical basis for A-formulas satisfiability checking - 2

The main theorem

Let φ be a A-formula. Then, φ is finitely satisfiable if and only if it is satisfiable on a model whose cardinality is strictly less than $2^m \cdot m + 1$, where m is the number of diamonds and boxes in φ.

The theoretical basis for A-formulas satisfiability checking - 2

The main theorem

Let φ be a A-formula. Then, φ is finitely satisfiable if and only if it is satisfiable on a model whose cardinality is strictly less than $2^m \cdot m + 1$, where m is the number of diamonds and boxes in φ.

Clearly, this result shows that A is NEXPTIME; in 2007 we also proved that it is NEPTIME-complete.

A-rules - 1

Given a A-formula φ, we want to establish if there exists a finite model $\mathbf{M}$ and an interval $[d_i, d_j]$ in it such that $\mathbf{M}, [d_i, d_j] \Vdash \varphi$.

Given a A-formula φ, we want to establish if there exists a finite model **M** and an interval $[d_i, d_j]$ in it such that $\mathbf{M}, [d_i, d_j] \Vdash \varphi$.

It is easy to see that, since A has no past operators, this is equivalent to ask whether $\mathbf{M}, [d_0, d_1] \Vdash \varphi$, being d_0, d_1 the first two points of the models (*initial satisfiability*).

A-rules - 1

Given a A-formula φ, we want to establish if there exists a finite model $\mathbf{M}$ and an interval $[d_i, d_j]$ in it such that $\mathbf{M}, [d_i, d_j] \Vdash \varphi$.

It is easy to see that, since A has no past operators, this is equivalent to ask whether $\mathbf{M}, [d_0, d_1] \Vdash \varphi$, being d_0, d_1 the first two points of the models (*initial satisfiability*). The classical operators are treated in the standard way. As for the modal operator, we use two rules, one to deal with diamonds and the other one to deal with boxes.

Given a A-formula φ, we want to establish if there exists a finite model **M** and an interval $[d_i, d_j]$ in it such that $\mathbf{M}, [d_i, d_j] \Vdash \varphi$.

It is easy to see that, since A has no past operators, this is equivalent to ask whether $\mathbf{M}, [d_0, d_1] \Vdash \varphi$, being d_0, d_1 the first two points of the models (*initial satisfiability*). The classical operators are treated in the standard way. As for the modal operator, we use two rules, one to deal with diamonds and the other one to deal with boxes.

A run of the tableau determines if a given formula φ is satisfiable over the interval $[d_0, d_1]$; thus, a *node* is a *labelled formula* of the type $\psi, [d_i, d_j]$.

A-rules - 2

The modal rules are:

$$(box)\ \frac{[A]\psi : [d_i, d_j]}{\psi : [d_j, d_{j+1}], \ldots, \psi : [d_j, d_N]},$$

A-rules - 2

The modal rules are:

$$(box)\ \frac{[A]\psi : [d_i, d_j]}{\psi : [d_j, d_{j+1}], \ldots, \psi : [d_j, d_N]},$$

$$(dia)\ \frac{\langle A\rangle\psi : [d_i, d_j]}{\psi : [d_j, d_{j+1}] \mid \ldots \mid \psi : [d_j, d_N] \mid \psi : [d_j, d'_j] \mid \ldots \mid \psi : [d_j, d'_N]},$$

A-rules - 2

The modal rules are:

$$(box)\ \frac{[A]\psi : [d_i, d_j]}{\psi : [d_j, d_{j+1}], \ldots, \psi : [d_j, d_N]},$$

$$(dia)\ \frac{\langle A\rangle\psi : [d_i, d_j]}{\psi : [d_j, d_{j+1}] \mid \ldots \mid \psi : [d_j, d_N] \mid \psi : [d_j, d'_j] \mid \ldots \mid \psi : [d_j, d'_N]},$$

where for each $j \leq h \leq N$, d_h is a point in D and d'_h is a new point added to D and placed immediately after d_h and immediately before d_{h+1} (when $h < N$).

So, every application of the classical $\vee$-rule a new branch is created, on which we keep trace of the current domain D.

A branch is declared *closed* in one of two situations: either we find a contradiction (i.e., both p and $\neg p$ are labeled with $[d_i, d_j]$ for some proposition p), or the cardinality of the domain D associated with the branch is too big (termination condition). Otherwise the branch is *open*, and rules are applied only to open branches.

Branch Managing

A branch is declared *closed* in one of two situations: either we find a contradiction (i.e., both p and $\neg p$ are labeled with $[d_i, d_j]$ for some proposition p), or the cardinality of the domain D associated with the branch is too big (termination condition). Otherwise the branch is *open*, and rules are applied only to open branches.

On a branch B, a labeled formula $\varphi, [d_i, d_j]$ is *inactive* if φ is a literal or if a rule has been already applied to it; otherwise it is *active*.

Branch Managing

A branch is declared *closed* in one of two situations: either we find a contradiction (i.e., both p and $\neg p$ are labeled with $[d_i, d_j]$ for some proposition p), or the cardinality of the domain D associated with the branch is too big (termination condition). Otherwise the branch is *open*, and rules are applied only to open branches.

On a branch B, a labeled formula $\varphi, [d_i, d_j]$ is *inactive* if φ is a literal or if a rule has been already applied to it; otherwise it is *active*.

Branch-expansion is performed top-down: the first active formula is found and its (only) applicable rule is applied to it. If no more active formulas are found on a open branch, then the system terminated with success (satisfiable formula); if all branches have been closed, then the result is negative (unsatisfiable formula).

Formula, Nodes, and Branches Representation

The input formula φ is supposed to be a logical conjunction of formulas, each one of which encoded in a different text line.

Formula, Nodes, and Branches Representation

The input formula φ is supposed to be a logical conjunction of formulas, each one of which encoded in a different text line.

The input formula is first transformed into its equivalent *negated normal form*, and stored into a *syntactic tree* (leaves are propositional letters, and internal nodes are Boolean or modal operators).

Formula, Nodes, and Branches Representation

The input formula φ is supposed to be a logical conjunction of formulas, each one of which encoded in a different text line.

The input formula is first transformed into its equivalent *negated normal form*, and stored into a *syntactic tree* (leaves are propositional letters, and internal nodes are Boolean or modal operators).

Nodes are represented by: a pointer to the subtree representing the formula labeling the node, two integer variables that identify the interval annotating the formula, and a Boolean flag (active or inactive).

Formula, Nodes, and Branches Representation

The input formula φ is supposed to be a logical conjunction of formulas, each one of which encoded in a different text line.

The input formula is first transformed into its equivalent *negated normal form*, and stored into a *syntactic tree* (leaves are propositional letters, and internal nodes are Boolean or modal operators).

Nodes are represented by: a pointer to the subtree representing the formula labeling the node, two integer variables that identify the interval annotating the formula, and a Boolean flag (active or inactive).

A *branch* B is 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.

Search Procedure

Branches are collected into a *priority queue*. Initially, the queue contains only one branch with only one node, that is, $\varphi, [d_0, d_1]$. Such a node is active, and the cardinality of the domain associated with the unique branch is 2.

Search Procedure

Branches are collected into a *priority queue*. Initially, the queue contains only one branch with only one node, that is, $\varphi, [d_0, d_1]$. Such a node is active, and the cardinality of the domain associated with the unique branch is 2.

The search procedure first selects the branch with highest priority; then, checks if it meets any closure condition, in which case the branch is closed and deleted, and the procedure re-starts; then, it selects the closest-to-the-root active node, and if there are none, the formula is declared satisfiable, being the current branch a model for it; finally, it applies the expansion rule to the selected node.

Search Procedure

Branches are collected into a *priority queue*. Initially, the queue contains only one branch with only one node, that is, $\varphi, [d_0, d_1]$. Such a node is active, and the cardinality of the domain associated with the unique branch is 2.

The search procedure first selects the branch with highest priority; then, checks if it meets any closure condition, in which case the branch is closed and deleted, and the procedure re-starts; then, it selects the closest-to-the-root active node, and if there are none, the formula is declared satisfiable, being the current branch a model for it; finally, it applies the expansion rule to the selected node.

If as a result of the search procedure all branches are deleted, the formula is declared unsatisfiable.

Priority Policies

To determine the priority of a branch in the queue, we implemented four different (and complete) policies:

Priority Policies

To determine the priority of a branch in the queue, we implemented four different (and complete) policies:

- FIFO: the standard first-in-first-out;
- LDF: largest domain branches are expanded first;
- SDF: smallest domain branches are expanded first;
- GAN: branches with greatest number of active nodes are expanded first;
- SDF: branches with smallest number of active nodes are expanded first.

Priority Policies

To determine the priority of a branch in the queue, we implemented four different (and complete) policies:

- FIFO: the standard first-in-first-out;
- LDF: largest domain branches are expanded first;
- SDF: smallest domain branches are expanded first;
- GAN: branches with greatest number of active nodes are expanded first;
- SDF: branches with smallest number of active nodes are expanded first.

The policy is decided before starting the search procedure and cannot be changed during a run.

Experimental Results - 1

We tested our implementation against a benchmark of problems that can be classified into two categories:

Experimental Results - 1

We tested our implementation against a benchmark of problems that can be classified into two categories:

Combinatorics: used to test the scalability of the procedure. 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 $[A]\neg(p_i \wedge p_j)$ $(i \neq j)$.

Experimental Results - 1

We tested our implementation against a benchmark of problems that can be classified into two categories:

Combinatorics: used to test the scalability of the procedure. 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 $[A]\neg(p_i \wedge p_j)$ $(i \neq j)$.

Randomized: a series of 36 completely random problems, generated in a suitable clausal form, to simulate the behaviour in real cases.

Experimental Results - 2

COMBINATORICS

n	Policy (sec)					Outcome
n	FIFO	SDF	LDF	SAN	GAN	(size)
1	0.004	0.004	0.004	0.004	0.004	4
2	0.004	0.008	0.004	0.004	0.008	5
3	0.008	0.15	0.03	0.008	0.03	6
4	0.01	–	30.07	0.01	30.29	7
5	0.012	–	–	0.012	–	8
6	0.02	–	–	0.03	–	9
7	0.07	–	–	0.07	–	10
8	0.15	–	–	0.16	–	11
9	0.3	–	–	0.32	–	12
10	0.56	–	–	0.59	–	13
11	0.99	–	–	1.06	–	14

n	Policy (sec)					Outcome
n	FIFO	SDF	LDF	SAN	GAN	(size)
12	1.67	–	–	1.79	–	15
13	2.73	–	–	2.94	–	16
14	4.25	–	–	4.55	–	17
15	6.56	–	–	7.08	–	18
16	9.77	–	–	10.82	–	19
17	14.42	–	–	15.40	–	20
18	20.79	–	–	22.20	–	21
19	29.28	–	–	32.11	–	22
20	40.91	–	–	44.09	–	23
21	–	–	–	–	–	–
22	–	–	–	–	–	–

Experimental Results - 3

RANDOMIZED

n	Policy (sec) FIFO	SDF	LDF	SAN	GAN	Outcome (size)
1	0.004	0.004	0.004	0.004	0.004	4
2	0.004	0.004	0.004	0.004	0.004	4
3	0.004	0.004	0.004	0.004	0.004	4
4	0.004	0.004	0.004	0.004	0.004	4
5	0.004	0.004	0.004	0.004	0.004	4
6	0.004	0.004	0.004	0.004	0.004	4
7	0.07	0.23	0.004	0.18	0.004	3 / 4
8	0.004	0.004	0.004	0.004	0.004	4
9	0.004	0.004	0.004	0.004	0.004	4
10	0.004	0.004	0.004	0.004	0.004	4
11	0.004	0.004	0.004	0.004	0.004	4
12	0.004	0.004	0.004	0.004	0.004	4
13	0.01	0.04	0.004	0.02	0.004	4
14	0.004	0.004	0.004	0.004	0.004	4
15	0.004	0.004	0.004	0.004	0.004	4
16	0.004	1.37	0.004	0.01	0.004	4
17	0.004	0.004	0.004	0.004	0.004	4
18	0.004	0.004	0.004	0.004	0.004	3

n	Policy (sec) FIFO	SDF	LDF	SAN	GAN	Outcome (size)
19	1.66	45.43	0.68	1.91	0.02	3 / 4
20	0.02	0.004	0.03	0.03	0.004	2 / 4
21	0.004	0.004	0.004	0.004	0.004	4
22	0.74	14.08	0.004	1.04	0.004	4
23	0.004	0.004	0.004	0.004	0.004	4
24	0.004	0.004	0.004	0.004	0.004	4
25	–	–	–	–	–	–
26	0.004	0.004	0.004	0.004	0.004	4
27	0.004	–	0.004	0.01	–	3 / 4
28	0.004	0.004	0.004	0.004	0.004	4
29	0.004	–	0.004	0.004	0.004	4
30	0.14	0.08	0.04	0.19	0.01	2 / 4
31	0.004	0.004	0.004	0.004	0.004	unsat
32	0.25	–	0.02	0.31	0.004	2 / 4
33	0.004	0.004	0.004	0.004	0.004	4
34	–	–	0.02	0.004	0.02	2 / 4
35	0.004	–	0.004	–	0.004	2 / 4
36	–	–	–	–	1.2	3

Conclusions

Ten years of research on interval-temporal logic have given a number of results concerning decidability, undecidability, expressive power and properties of more and more interesting languages.

Conclusions

Ten years of research on interval-temporal logic have given a number of results concerning decidability, undecidability, expressive power and properties of more and more interesting languages.

Unfortunately, even for simple decidable cases, not much effort has been devoted to the implementation and realization of the theoretical procedures. We know that there are expressive enough and relatively computationally simple interval-based languages that can be used in many practical applications, but the gap with the reality of the implementations is still to be filled.

Conclusions

Ten years of research on interval-temporal logic have given a number of results concerning decidability, undecidability, expressive power and properties of more and more interesting languages.

Unfortunately, even for simple decidable cases, not much effort has been devoted to the implementation and realization of the theoretical procedures. We know that there are expressive enough and relatively computationally simple interval-based languages that can be used in many practical applications, but the gap with the reality of the implementations is still to be filled.

This work represents a first step in this direction. On the web page *http://www.di.unisa.it/dottorandi/dario.dellamonica/tableaux/* it is possible to find the system available for testing. We plan to expand and improved it to deal with more expressive languages and with different classes of linear orders. Moreover, we also plan to solve a new problem that we found during our experiments: there is no literature available on the task of generating a proper benchmark in the interval case.