And-Or Tableaux for Fixpoint Logics with Converse: LTL, CTL, PDL, CPDL

Rajeev Goré
(Pietro Abate, Ling Anh Nguyen, Linda Postniece, Jimmy Thomson, Florian Widmann)

Logic and Computation Group
College of Engineering and Computer Science
The Australian National University
rajeev.gore@anu.edu.au

IJCAR 2014 Vienna

A Motivating Example from Business Process Modelling

Banking: rules for ensuring processes meet anti-fraud legislation

Example: rules for granting a request to open a bank account

- Rule 1: A risk assessment (ra) must eventually be carried out for each request to open a bank account (ro)
- Rule 2: A request to open a bank account (ro) is granted (rog) only if the risk is assessed as low (ral)
- Rule 3: A due diligence assessment (dd) must eventually be carried out for each request to open a bank account (ro)
- Rule 4: If a person fails due diligence (ddf) then he or she must be blacklisted (bl)

Question: how to check these rules for consistency and sanity?

A Motivating Example from Business Process Modelling

Banking: rules for ensuring processes meet anti-fraud legislation

Example: rules for granting a request to open a bank account

- Rule 1: A risk assessment (ra) must eventually be carried out for each request to open a bank account (ro)
- Rule 2: A request to open a bank account (ro) is granted (rog) only if the risk is assessed as low (ral)
- Rule 3: A due diligence assessment (dd) must eventually be carried out for each request to open a bank account (ro)
- Rule 4: If a person fails due diligence (ddf) then he or she must be blacklisted (bl)

Question: how to check these rules for consistency and sanity?

Need: a logic which captures temporal notions like "eventually", "after", "before" as well as "if then", "only if" etc.

Fixpoint Logics: Linear Temporal Logic

Syntax: CPL plus "next", "until", "before", "always", "eventually"

$$
\begin{array}{rcl}
\mathit{atom} & ::= & p \mid q \mid r \mid \cdots \\
\varphi, \psi & ::= & \mathit{atom} \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \to \psi \\
 & & \mid \bigcirc\varphi \mid \varphi\,\mathcal{U}\,\psi \mid \varphi\,\mathcal{B}\,\psi \mid [F]\,\varphi \mid \langle F\rangle\,\varphi
\end{array}
$$

Fixpoint Logics: Linear Temporal Logic

Syntax: CPL plus "next", "until", "before", "always", "eventually"

$$
\begin{array}{rcl}
\mathit{atom} & ::= & p \mid q \mid r \mid \cdots \\
\varphi, \psi & ::= & \mathit{atom} \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \to \psi \\
 & & \mid \bigcirc\varphi \mid \varphi\,\mathcal{U}\,\psi \mid \varphi\,\mathcal{B}\,\psi \mid [F]\,\varphi \mid \langle F\rangle\,\varphi
\end{array}
$$

Semantics: infinite linear discrete sequence of points $s_0, s_1, s_2, \cdots$

Model: truth value **t** exor **f** to each atomic formula at each state s_i

Evaluate CPL formula: at state s_i using truth tables

Evaluate "temporal" formula: using relative order of states

Fixpoint Logics: Linear Temporal Logic

Syntax: CPL plus "next", "until", "before", "always", "eventually"

$$
\begin{array}{rcl}
atom & ::= & p \mid q \mid r \mid \cdots \\
\varphi, \psi & ::= & atom \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \rightarrow \psi \\
 & & \mid \bigcirc\varphi \mid \varphi\,\mathcal{U}\,\psi \mid \varphi\,\mathcal{B}\,\psi \mid [F]\,\varphi \mid \langle F\rangle\,\varphi
\end{array}
$$

Semantics: infinite linear discrete sequence of points $s_0, s_1, s_2, \cdots$

Model: truth value **t** exor **f** to each atomic formula at each state s_i

Evaluate CPL formula: at state s_i using truth tables

Evaluate "temporal" formula: using relative order of states

Satisfiable: true at some state in some sequence

Valid: true in all states of every sequence

Lemma: φ is valid if and only if $\neg\varphi$ is unsatisfiable

LTL: Kripke Semantics for Assigning $\mathbf{t}$ at State s_i

Connective	s_i	s_{i+1}	$\cdots$	s_j	$\cdots$	s_k	s_{k+1}	$\cdots$
next	$\bigcirc p$	p						
$\mathcal{U}$*ntil*	$p\,\mathcal{U}\,q$	p	$\cdots$	p	$\cdots$	q		
$\mathcal{B}$*efore*	$p\,\mathcal{B}\,q, \neg q$	$\neg q$	$\cdots$	$\neg q$	$\cdots$	$\neg q$	$\neg q$	$\cdots$
$\mathcal{B}$*efore*	$p\,\mathcal{B}\,q, \neg q$	$\neg q$	$\cdots$	$p, \neg q$	$\cdots$	q		

- $\mathcal{U}$: is "strong" and demands that eventually q is true at some s_k
- $\mathcal{U}$: s_k is the first state after or equal to s_i where q is true
- $\mathcal{B}$: is "weak" since it does not demand eventually q
- $\mathcal{B}$: is "strictly before" since q at s_i is forbidden

LTL: Defined Connectives

Defn	s_i	s_{i+1}	s_{i+2}	s_j	$\cdots$	s_k
$\top := (p \vee \neg p)$	$\top$	$\top$	$\top$	$\top$	$\cdots$	$\top$
$\bot := \neg\top$	*never true*					
$\langle F \rangle \psi := (\top \,\mathcal{U}\, \psi)$	$\langle F \rangle \psi$	$\top$	$\top$	$\top$	$\cdots$	ψ
	$\neg \langle F \rangle \psi$	$\neg\psi$	$\neg\psi$	$\neg\psi$	$\cdots$	$\neg\psi$
$[F]\, \varphi := (\neg \langle F \rangle \neg\varphi)$	$[F]\, \varphi, \varphi$	φ	$\cdots$	φ	$\cdots$	φ

$\langle F \rangle \psi$: captures "eventually in the Future ψ"

$[F]\, \varphi$: captures "always in the Future φ"

Encoding Our Example from Business Process Modelling

Rule 1: A risk assessment (ra) must eventually be carried out for each request to open a bank account (ro)

$$[F]\,(ro \rightarrow \langle F \rangle\, ra)$$

Rule 2: A request to open a bank account (ro) is granted (rog) only if the risk is low (ral)

$$[F]\,(rog \rightarrow ral)$$

Rule 3: A due diligence assessment (dd) must eventually be carried out for each request to open a bank account (ro)

$$[F]\,(ro \rightarrow \langle F \rangle\, dd)$$

Rule 4: If a person fails due diligence (ddf) then he or she must be blacklisted (bl)

$$[F]\,((dd \wedge ddf) \rightarrow \langle F \rangle\, bl)$$

Awad et al: An iterative approach to synthesize business process templates from compliance rules. Inf. Syst. 37(8): 714-736 (2012)

Fixpoint Logics: PLTL, CTL, LCK, PDL, CTL*, μ-calculus

PLTL: $\varphi \, \mathcal{U} \, \psi \qquad \leftrightarrow \psi \vee \bigcirc(\varphi \, \mathcal{U} \, \psi)$

CTL: $E\,(\varphi \, \mathcal{U} \, \psi) \qquad \leftrightarrow \psi \vee EXE(\varphi \, \mathcal{U} \, \psi)$

LCK: $\langle C \rangle \psi \qquad \leftrightarrow \langle E \rangle \psi \vee \langle E \rangle \langle C \rangle \psi$

PDL: $\langle \alpha * \rangle \psi \qquad \leftrightarrow \psi \vee \langle \alpha \rangle \langle \alpha * \rangle \psi$

Kripke semantics: involve a base "step-relation" e.g. "next"

Least Fixpoints: "now or after one step"

$\langle F \rangle \, \varphi$: is least fixpoint $\mu Z.(\varphi \vee \bigcirc Z)$

$[F] \, \varphi$: is greatest fixpoint $\nu Z.(\varphi \wedge \bigcirc Z)$

Standard Translation: none are first-order definable

Fragments: of monadic second order logic of one successor S1S

The main logical problems and their complexity

Given:

- a logic L $\in \{LTL, CTL, PDL, \ldots\}$
- a finite set $\mathcal{T}$ of global assumption (TBox) L-formulae
- a single L-formula φ

Satisfiability: is there an L-model that makes $\mathcal{T}$ true everywhere and makes φ true somewhere

Global logical consequence: for every L-model M, if M makes $\mathcal{T}$ true everywhere then M makes φ true everywhere (glc)

Validity: put $\mathcal{T} = \emptyset$

Counter-models: if φ not a glc of $\mathcal{T}$ then output a model which makes $\mathcal{T}$ true everywhere and φ false somewhere

Complexity: deciding glc is $\textsc{exptime}$-complete

Why Tableaux?

Tableaux algorithms provide some of the most efficient methods for automated reasoning in non-classical logics

Long history (1979-2010) of attempts to devise tableau algorithms for fixpoint logics without much success

This talks summarises almost seven years of work from 2003-2010 with various people

Modal Tableaux as And-Or Trees (Using NNF)

(id) $\dfrac{p; \neg p; X}{}$ $\qquad$ $(\wedge)$ $\dfrac{\varphi \wedge \psi; X}{\varphi; \psi; X}$ $\qquad$ $(\vee)$ $\dfrac{\varphi \vee \psi; X}{\varphi; X \mid \psi; X}$

$$(\mathrm{K})\ \frac{\langle\rangle\,\varphi_1\ ;\ \ \langle\rangle\,\varphi_2\ ;\ \ \cdots\ ;\ \ \langle\rangle\,\varphi_k\ ;\ \ []\,X\ ;\ \ Z}{\mathcal{T}; \varphi_1; X\ \ ||\ \ \mathcal{T}; \varphi_2; X\ \ ||\ \ \cdots\ \ ||\ \ \mathcal{T}; \varphi_k; X}\ \dagger$$

$\mathcal{T}$ is a given finite set of global assumption formulae

X, Y, Z are finite possibly empty sets of formulae

$\varphi; X$ stands for a partition of the non-empty set $\{\varphi\} \cup X$

$\dagger$: Z is saturated (contains no top level $\wedge, \vee, []$ formulae)

A K-tableau for Y given global assumptions $\mathcal{T}$

is an inverted (or-branching) tree of nodes with:

1. a root node `nnf` $(\mathcal{T}; Y)$
2. and such that all children nodes are obtained from their parent node by instantiating a rule of inference

Propagation: Determining the status of nodes

$$\text{(id)}\ \frac{p; \neg p; X}{} \qquad (\wedge)\ \frac{\varphi \wedge \psi; X}{\varphi; \psi; X} \qquad (\vee)\ \frac{\varphi \vee \psi; X}{\varphi; X \mid \psi; X}$$

$$\text{(K)}\ \frac{\langle\rangle\, \varphi_1\ ;\ \ \langle\rangle\, \varphi_2\ ;\ \ \cdots\ ;\ \ \langle\rangle\, \varphi_k\ ;\ \ []\, X\ ;\ \ Z}{\mathcal{T}; \varphi_1; X\ \ ||\ \ \mathcal{T}; \varphi_2; X\ \ ||\ \ \cdots\ \ ||\ \ \mathcal{T}; \varphi_k; X}\ \dagger$$

Status of every node is initially `unexpanded`

Status `sat` if we can apply no rule

Status `unsat` if we apply (id)

Status changes to `open` for other rule applications

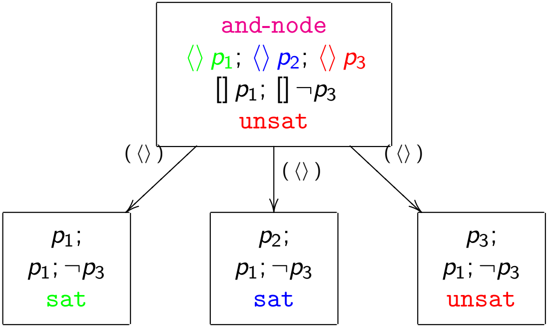
`or-node`: becomes `sat` if any child becomes `sat` and becomes `unsat` if all children become `unsat`

`and-node`: becomes `unsat` if any child becomes `unsat` and becomes `sat` if all children become `sat`

Example: the And-branching (K)-rule

$$(\mathrm{K})\ \frac{\langle\rangle\,\varphi_1\ ;\ \ \langle\rangle\,\varphi_2\ ;\ \cdots\ ;\ \ \langle\rangle\,\varphi_k\ ;\ \ []\,X; Z}{\mathcal{T}; \varphi_1; X \ \ || \ \ \mathcal{T}; \varphi_2; X \ \ || \ \ \cdots \ \ || \ \ \mathcal{T}; \varphi_k; X}\ \dagger\ (Z \text{ saturated})$$

Only one and-or tableau for $\langle\rangle\, p_1;\ \langle\rangle\, p_2;\ \langle\rangle\, p_3;\ []\, p_1;\ []\, \neg p_3$



Example: Termination Requires Ancestor Loops

$$(\mathrm{K})\ \frac{\langle\rangle\,\varphi_1\ ;\ \ \langle\rangle\,\varphi_2\ ;\ \cdots\ ;\ \ \langle\rangle\,\varphi_k\ ;\ \ []\,X;Z}{\mathcal{T};\varphi_1;X\ \ \ ||\ \ \ \mathcal{T};\varphi_2;X\ \ \ ||\ \ \ \cdots\ \ \ ||\ \ \ \mathcal{T};\varphi_k;X}\ \dagger\ (Z \text{ saturated})$$

The tableau for $\mathcal{T} = \{\langle\rangle\,p\}$ and $\varphi := q$ loops!

$$\begin{tikzcd}
\boxed{\langle\rangle\,p\ ;\ q} \arrow[d, "\langle\rangle"] \\
\boxed{p\ ;\ \langle\rangle\,p} \arrow[d] \\
\boxed{p\ ;\ \langle\rangle\,p}
\end{tikzcd}$$

Solution: check whether new node exists already on the current branch and stop when we see the same node (loop check)

Note: looping nodes return `open` ... turns to `sat` at end of algorithm

Soundness, Completeness and Termination

Thm: every and-or tableau (with ancestor loops) is finite

Thm: any and-or tableau for $\mathcal{T} \cup \{\neg\varphi_0\}$ returns `unsat` iff φ_0 is glc of $\mathcal{T}$

Complexity:: Worst-case is is 2EXPTIME i.e. $O(2^{2^n})$

Cause: tree tableaux can explore the same node on multiple branches

Optimisations: practical implementations use many optimisations

Tableau Rules for LTL in Smullyan's $\alpha-$ and $\beta-$notation

Notation captures abstract "conjunctive" and "disjunctive" notions

$\boldsymbol{\alpha}$	$\boldsymbol{\alpha}_1$	$\boldsymbol{\alpha}_2$
$\varphi \land \psi$	φ	ψ
$\varphi\, B\, \psi$	$\sim\psi$	$\varphi \lor \bigcirc(\varphi\, B\, \psi)$
$[F]\,\varphi$	φ	$\bigcirc\,[F]\,\varphi$

$\boldsymbol{\beta}$	$\boldsymbol{\beta}_1$	$\boldsymbol{\beta}_2$
$\varphi \lor \psi$	φ	ψ
$\varphi\, U\, \psi$	ψ	$\varphi \land \bigcirc(\varphi\, U\, \psi)$
$\langle F\rangle\,\varphi$	φ	$\bigcirc\,\langle F\rangle\,\varphi$

Define: $\sim\psi := NNF(\neg\psi)$

Prop: all instances of $\boldsymbol{\alpha} \leftrightarrow \boldsymbol{\alpha}_1 \land \boldsymbol{\alpha}_2$ and $\boldsymbol{\beta} \leftrightarrow \boldsymbol{\beta}_1 \lor \boldsymbol{\beta}_2$ are valid

Assume: that all formulae are in Negation Normal Form

Tableau Rules: only applicable to non-starred formulae

$$(\boldsymbol{\alpha})\ \frac{\Gamma\,;\ \boldsymbol{\alpha}}{\boldsymbol{\alpha}^*\,;\ \Gamma\,;\ \boldsymbol{\alpha}_1\,;\ \boldsymbol{\alpha}_2} \qquad (\boldsymbol{\beta})\ \frac{\Gamma\,;\ \boldsymbol{\beta}}{\boldsymbol{\beta}^*\,;\ \Gamma\,;\ \boldsymbol{\beta}_1 \mid \boldsymbol{\beta}^*\,;\ \Gamma\,;\ \boldsymbol{\beta}_2} \qquad (\bigcirc)\ \frac{\Lambda\,;\ \bigcirc\varphi\,;\ \bigcirc\Delta}{\varphi\,;\ \Delta}$$

where Λ contains only atoms, negated atoms and starred formulae

Traditional Multipass Graph Methods (1979-1998)

Phase 1: apply rules to obtain a cyclic graph

Phase 2: using multiple passes, prune inconsistent nodes and nodes that contain eventualities unfulfilled by the current graph

Eventuality: formulae which can be postponed for ever

LTL eventualities: $\varphi \mathcal{U} \psi$ and $\langle F \rangle \varphi := (\top \mathcal{U} \varphi)$

$$
\begin{tikzcd}
 & \boxed{\begin{array}{c}\texttt{or-node}\\ \varphi \mathcal{U} \psi \ ; \ X\end{array}} \arrow[dl, "\textit{fulfill now}"'] \arrow[dr, "\textit{procrastinate}"] & \\
\boxed{\psi \ ; \ X} & & \boxed{\varphi \ ; \ \bigcirc(\varphi \mathcal{U} \psi) \ ; \ X}
\end{tikzcd}
$$

Answer `unsat`: if root node gets pruned

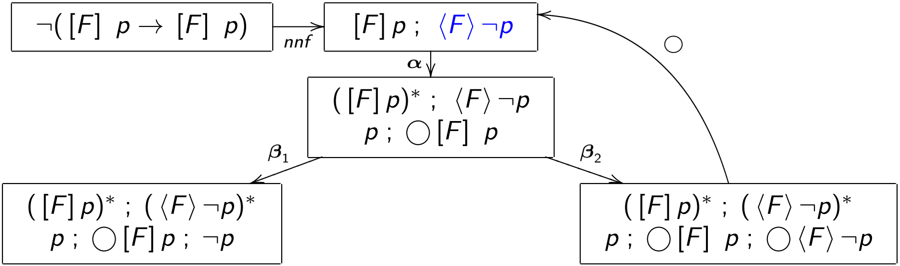
Answer `sat`: if no pruning possible (all eventualities fulfilled)

Example: is $[F]\,p \rightarrow [F]\,p$ LTL-valid ? Phase 1

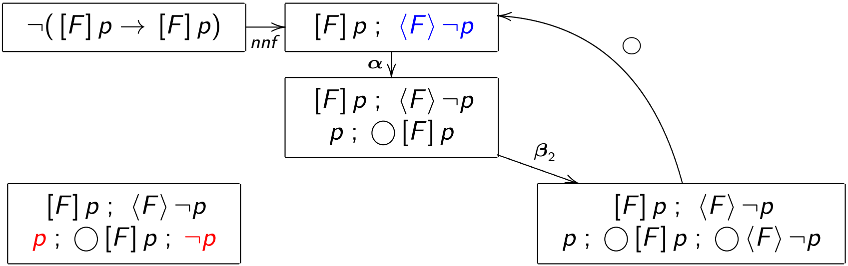
$\boldsymbol{\alpha}$	$\boldsymbol{\alpha}_1$	$\boldsymbol{\alpha}_2$
$[F]\ \varphi$	φ	$\bigcirc[F]\ \varphi$

$\boldsymbol{\beta}$	$\boldsymbol{\beta}_1$	$\boldsymbol{\beta}_2$
$\langle F\rangle\,\varphi$	φ	$\bigcirc\langle F\rangle\,\varphi$

$$(\bigcirc)\ \frac{\Lambda\,;\ \bigcirc\varphi\,;\ \bigcirc\Delta}{\varphi\,;\ \Delta}\,\dagger$$



Example: is $[F]\,p \to [F]\,p$ LTL-valid ? Phase 2



Unstar all starred formulae

Prune the node containing $\{p, \neg p\}$

Prune the root containing $\langle F\rangle\,\neg p$ since no path reaches $\neg p$

That is, $[F]\,p\,;\ \langle F\rangle\,\neg p$ is not LTL-satisfiable.

Hence $[F]\,p \to [F]\,p$ is LTL-valid.

Properties of the graph multipass tableau method

Worst case complexity: is EXPTIME

Best case complexity: can be better than EXPTIME

Disadvantage: can do unnecessary work

Example: $\langle F \rangle \varphi \wedge \langle F \rangle \Psi$ where Ψ is huge but irrelevant

Ideal: build, prune and check eventualities "on the fly"

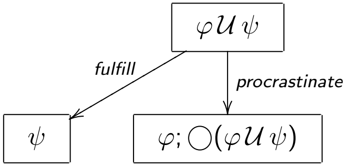
One-pass And-Or Tree Tableaux for LTL (1998)

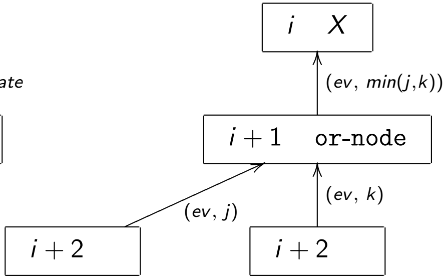
Use the tree-tableaux for modal logics, with ancestor loops

One pass: build a rooted and-or tree with ancestor cycles and allow nodes to pass up a list of unfulfilled eventualities

Advantage: can explore one branch at a time

Disadvantage: suboptimal (2EXPTIME)



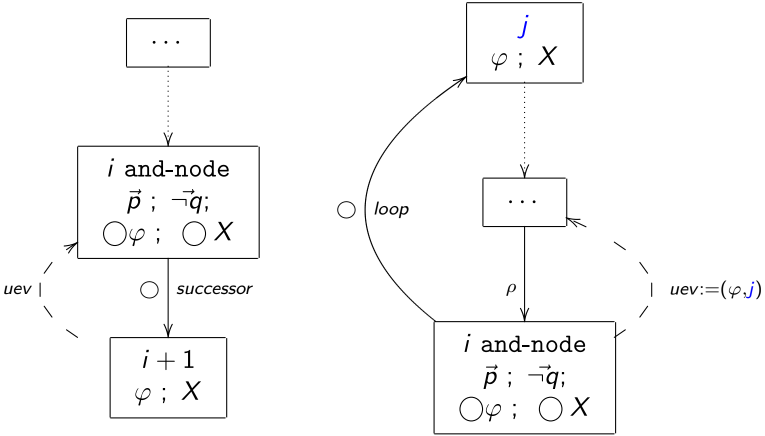


Close: any node at height h with an eventuality $(ev, h+1)$

S. Schwendimann. A new one-pass tableau calculus for PLTL. In *Proc. TABLEAUX'98*, LNAI 1397:277-291. Springer, 1998.

One-pass And-Or Tree Tableaux

Use the tree-tableaux for modal logics, with ancestor loops

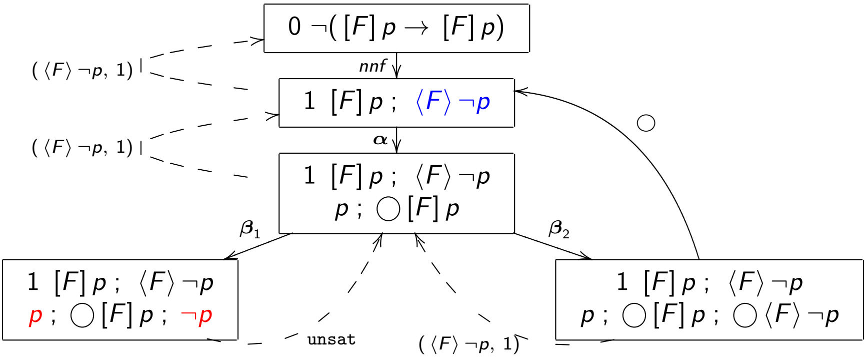


$\bigcirc$ rule: either creates a successor or blocks on an existing ancestor

New: rules now pass back a set of pairs (ev, j) listing blocks

Tree-tableau method for LTL

Use the tree-tableaux for modal logics, with ancestor loops:

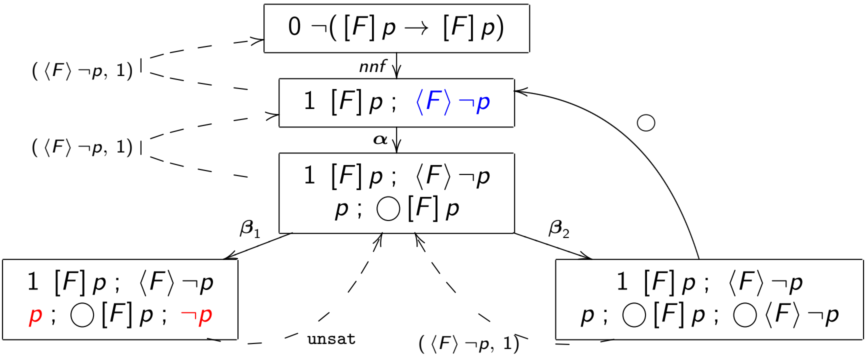


Return: extra information from children to parents

$(\langle F\rangle\neg p, j)$: eventualities that are blocked from being fulfilled and the height of their blocking ancestor exor unsat to indicate "closed"

Tree-tableau method for LTL

Use the tree-tableaux for modal logics, with ancestor loops:



Closure: node at level i receives back a pair (ev, j) where $j > i$

Hence: node 0 closes itself, thus closing the tableau

Reason: $\langle F \rangle \neg p$ not fulfilled by the subtree rooted at node 1

Properties of one-pass tree tableaux for LTL

Thm: φ is glc of $\mathcal{T}$ iff any one-pass tree-tableau for $\mathcal{T} \cup \neg\varphi$ closes

Complexity worst-case: 2EXPTIME close all branches

Complexity best-case: less first branch is open

Space: worst-case is EXPSPACE depth-first search

Advantage: may avoid unnecessary work

Example: $\langle F \rangle \varphi \wedge \langle F \rangle \Psi$ where Ψ is huge but irrelevant

Properties of one-pass tree tableaux for LTL

Thm: φ is glc of $\mathcal{T}$ iff any one-pass tree-tableau for $\mathcal{T} \cup \neg\varphi$ closes

Complexity worst-case: 2EXPTIME — close all branches

Complexity best-case: less — first branch is open

Space: worst-case is EXPSPACE — depth-first search

Advantage: may avoid unnecessary work

Example: $\langle F \rangle \varphi \wedge \langle F \rangle \Psi$ where Ψ is huge but irrelevant

What about CTL and PDL?
How to regain optimality?

Extensions to handle CTL and PDL

Replace $(\bigcirc)\ \dfrac{\bigcirc\varphi;\bigcirc X;\Lambda}{\varphi;X}\ \dagger$ by appropriate rules

CTL: models use branching-time

$$(\mathrm{EX})\ \frac{EX\varphi_1\ ;\ EX\varphi_2\ ;\ \cdots\ ;\ EX\varphi_k\ ;\ AX\ Y\ ;\ \Lambda}{\varphi_1\ ;\ Y\ ;\ \mathcal{T}\ \ ||\ \ \varphi_2\ ;\ Y\ ;\ \mathcal{T}\ \ ||\ \ \cdots\ \ ||\ \ \varphi_k\ ;\ Y\ ;\ \mathcal{T}}\ \dagger$$

Eventualities: now there are two

$E(\varphi\,\mathcal{U}\,\psi)$: β-rule uses $min(i,j)$ to track "longest" loop as for LTL

$A(\varphi\,\mathcal{U}\,\psi)$: β-rule uses $max(i,j)$ to track "shortest" loop

Extensions to handle CTL and PDL

Replace $(\bigcirc)\ \dfrac{\bigcirc\varphi; \bigcirc X; \Lambda}{\varphi; X}\ \dagger$ by appropriate rules

CTL: models use branching-time

$$(\mathrm{EX})\ \frac{EX\varphi_1\ ;\ EX\varphi_2\ ;\ \cdots\ ;\ EX\varphi_k\ ;\ AX\ Y\ ;\ \Lambda}{\varphi_1\ ;\ Y\ ;\ \mathcal{T}\ \|\ \varphi_2\ ;\ Y\ ;\ \mathcal{T}\ \|\ \cdots\ \|\ \varphi_k\ ;\ Y\ ;\ \mathcal{T}}\ \dagger$$

Eventualities: now there are two

$E(\varphi\,\mathcal{U}\,\psi)$: β-rule uses $min(i,j)$ to track "longest" loop as for LTL

$A(\varphi\,\mathcal{U}\,\psi)$: β-rule uses $max(i,j)$ to track "shortest" loop

PDL: branching and-rule but only eventuality is $\langle\alpha*\rangle\varphi$

regular expressions tracked using extra annotations in each node

Complexity: Still suboptimal ... how to regain optimality?

One-Pass Tableaux for Computation Tree Logic. LPAR 2007
An On-the-fly Tableau-based Decision Procedure for PDL-Satisfiability. M4M 2007

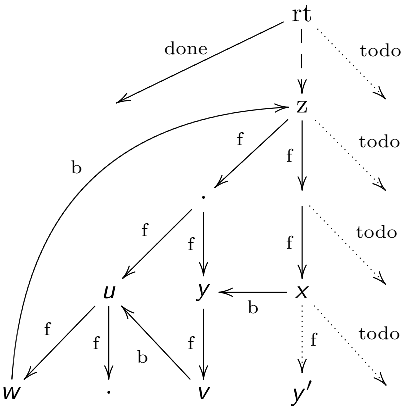
On-the-fly And-Or Graph Tableaux for LTL and PDL

Interleave: the graph building and graph pruning phases

Global Caching: allow cross branch edges to previous node copies

Advantage: complexity optimal

Disadvantage: requires more memory

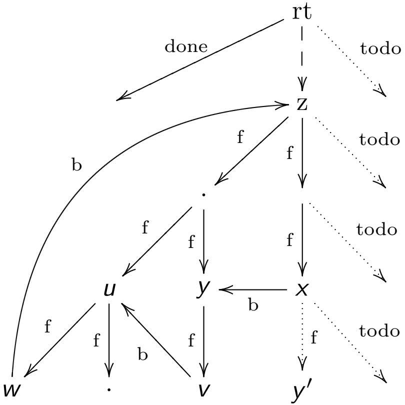


On-the-fly And-Or Graph Tableaux for LTL and PDL

Fulfilled: in the graph to the left of the frontier

Potentially fulfillable: path from x always procrastinates but hits a forward-ancestor of x on the current branch
e.g. the path x, y, v, u, w, z potential rescuers

Closed: unfulfilled and unfulfillable



Theorems for LTL and PDL only

Thm: root.status $=$ $\texttt{unsat}$ implies $\Gamma \cup \{\neg\varphi\}$ is L-unsatisfiable

Proof: not trivial

Thm: root.status $=$ $\texttt{sat}$ implies $\Gamma \cup \{\neg\varphi\}$ is L-satisfiable

Proof: not trivial

Thm: Worst-case complexity is EXPTIME

Proof: in worst case, we explore 2^n different (annotated) nodes

To Do: we could not find a way to handle the $A(\varphi \mathcal{U} \psi)$ eventuality of CTL using on-the-fly and-or graph tableaux

An Optimal On-the-Fly Tableau-Based Decision Procedure for PDL-Satisfiability. CADE 2009

Handling Converse in Modal And-Or Tableaux

Let us go right back to and-or modal tableaux and add converse

$$(\mathrm{id})\ \frac{p; \neg p; X}{} \qquad (\wedge)\ \frac{\varphi \wedge \psi; X}{\varphi; \psi; X} \qquad (\vee)\ \frac{\varphi \vee \psi; X}{\varphi; X \mid \psi; X}$$

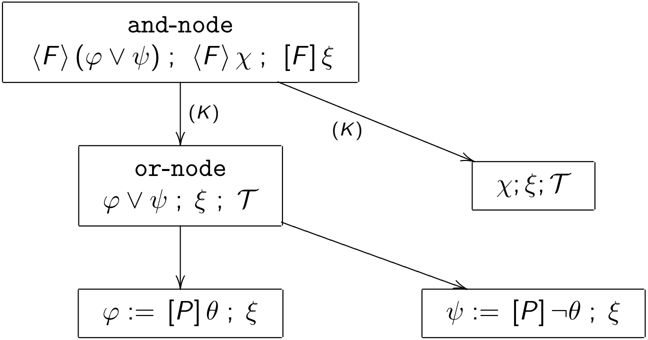
$$(\mathrm{K})\ \frac{\langle\rangle\,\varphi_1;\ \langle\rangle\,\varphi_2; \cdots ;\ \langle\rangle\,\varphi_k;\ []\,X; Z}{\mathcal{T}; \varphi_1; X \mid\mid \mathcal{T}; \varphi_2; X \mid\mid \cdots \mid\mid \mathcal{T}; \varphi_k; X}\ \dagger$$

Converse Operator Can Cause Incompatibility

Converse: add box and diamond connectives w.r.t. R^{-1}

Syntax: two copies $[F]$ for Future and $[P]$ for Past of $[]$

Axioms: $\varphi \to [F]\,\langle P\rangle\,\varphi \qquad \varphi \to [P]\,\langle F\rangle\,\varphi$



Usual solution: "restart", requires "dynamic blocking" and is suboptimal

Global Caching Not Possible

Node with no top-level conjunctions or disjunctions is a state

Non-states are pre-states

$$
\begin{tikzcd}
\text{states} & \boxed{\langle F\rangle \left(\bot \vee [P]\, q\right)} \arrow[d, "\langle F\rangle"'] & \surd & \boxed{\neg q,\ \ \langle F\rangle \left(\bot \vee [P]\, q\right)} \arrow[dll, "\langle F\rangle"'] & \times \\
\text{pre-state} & \boxed{\bot \vee [P]\, q\ ;\ \mathcal{T}} & \surd & &
\end{tikzcd}
$$

Satisfiability of pre-state does not imply satisfiability of state

Our Alternative Method

Starting point: "Sound Global Caching for $\mathcal{ALC}$" from DL07

Node contents fixed: at its creation $\rightsquigarrow$ no dynamic blocking

New status type: unexpanded, sat, unsat, open, *toosmall*

Dynamic Status: status of a node changes during the algorithm:

- Example 1: unexpanded $\rightsquigarrow$ unsat
- Example 2: unexpanded $\rightsquigarrow$ open $\rightsquigarrow$ toosmall

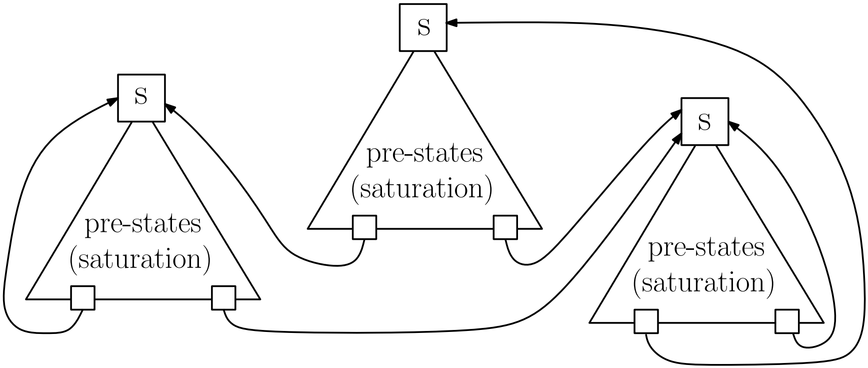
But: status of sat, unsat, or toosmall will not change

Global State Caching: never explore the same state again

Global caching: no two *nodes* with the same set of formulae

Global state caching: no two *states* with the same set of formulae

Saturation: apply $\wedge$ and $\vee$ rules until not applicable (giving state)



Special Nodes

Basic idea: separate a state from the saturation phase of another state.

state s

$$\begin{tikzcd} \boxed{\langle F\rangle\,\varphi \ ; \ [F]\,\Delta \ ; \ \cdots} \arrow[d, "R"'] \\ \boxed{\varphi \ ; \ \Delta \ ; \ \mathcal{T}} \arrow[d, dotted, "\text{saturate}"] \\ \boxed{[P]\,\Gamma \ ; \ \Sigma} \end{tikzcd}$$

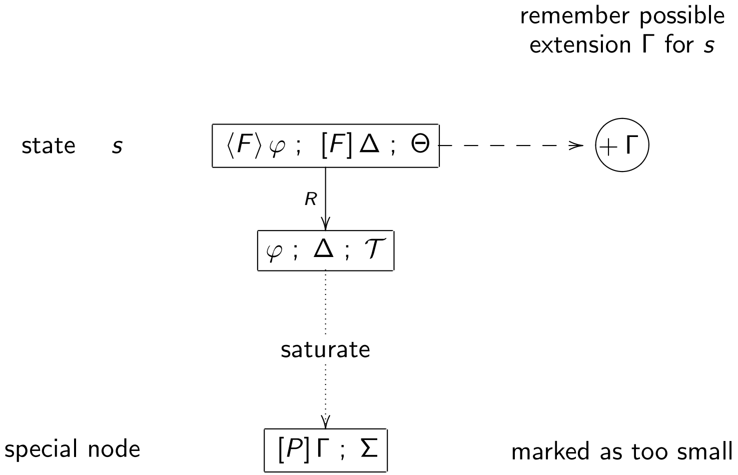
special node

If Γ in s, the special node is *compatible* with s.

Compatible Special Nodes

$$\begin{tikzcd}
\text{state} & \boxed{\langle F\rangle\,\varphi \; ; \; [F]\,\Delta \; ; \; \cdots \; ; \; \Gamma} \arrow[d, "R"'] \\
 & \boxed{\varphi \; ; \; \Delta \; ; \; \mathcal{T}} \arrow[d, dotted, "\text{saturate}"] \\
\text{special node} & \boxed{[P]\,\Gamma \; ; \; \Sigma} \arrow[d] \\
\begin{array}{c}\text{state}\\ \text{(might already exist)}\end{array} & \boxed{[P]\,\Gamma \; ; \; \Sigma}
\end{tikzcd}$$

Not Compatible Special Nodes



The Algorithm: Main Loop

- Pick node x which has not been expanded yet.
- Expand x, that is create children if needed and link them appropriately.
- Determine and set the status of x from unexpanded to either open, sat, unsat, or too small (see next slides).
- Explicit update/propagation phase, activated by status change of a node

Determining the Status of And and Or Nodes

Or-nodes

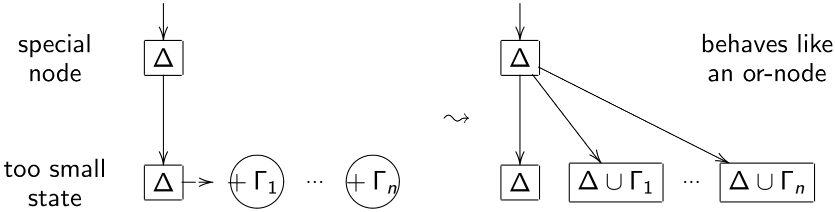
- some child is sat $\rightsquigarrow$ or-node is sat
- else: some child is open or unexpanded $\rightsquigarrow$ or-node is open
- else: some child is too small $\rightsquigarrow$ or-node is too small
- else: all children unsat $\rightsquigarrow$ or-node is unsat

And-nodes

- some child is unsat $\rightsquigarrow$ state is unsat
- else: some child is too small $\rightsquigarrow$ state is too small
- else: some child is open or unexpanded $\rightsquigarrow$ state is open
- else: all children are sat $\rightsquigarrow$ state is sat

Determining the Status of a Special Node

- child $\neq$ too small $\rightsquigarrow$ special node gets the same status
- state child too small with possible extensions $\Gamma_1, \ldots, \Gamma_n$:



Theorems for Converse Roles

Complexity: the algorithm terminates and runs in EXPTIME

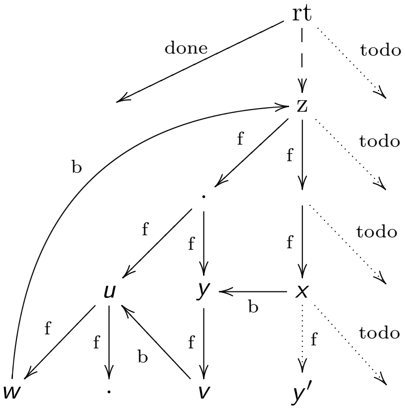
Thm: If a node is sat or remains open, its formulae are jointly satisfiable.

Thm: If a node is unsat, its formulae are not jointly satisfiable.

Thm: If the root node is too small, its formulae are not jointly satisfiable.

Sound Global State Caching for ALC with Inverse Roles.
TABLEAUX 2009

On-the-fly Complexity Optimal Tableaux for CPDL



On-the-fly Complexity Optimal Tableaux for CPDL

potential rescuers: on-the-fly graph tableaux

annotations: inside nodes to track regular expressions

special nodes: to handle converse

global state caching: to handle converse

On-the-fly Complexity Optimal Tableaux for CPDL

potential rescuers: on-the-fly graph tableaux

annotations: inside nodes to track regular expressions

special nodes: to handle converse

global state caching: to handle converse

`unexp`: initially all nodes have status unexpanded

`undef`: expanded but status not known

$\texttt{closed}(\mathit{alt})$: toosmall so needs to be cloned into alternatives

$\texttt{open}(\mathrm{prs}, \mathit{alt})$: open with potential rescuers, and alternatives in case it is closed later on

$\texttt{closed}(.)$: unsatisfiable, will never change again

idx_x: unique time stamp of when it became "defined" for proofs by induction

On-the-fly Complexity Optimal Tableaux for CPDL

Apply the following “rules” repeatedly in this order

Rule 1: picks an unexpanded node and expands it

Rule 2: picks an expanded but undefined node and computes its (initial) status. It also sets the correct time stamp.

Rule 3: picks an open node whose status has changed and recomputes its status.

Rule 4: is only applicable if all nodes are up-to-date. It picks an open node containing an eventuality φ which is currently not fulfilled in the graph and which does not have any potential rescuers either. As this indicates that φ can never be fulfilled, the node is closed.

On-the-fly Complexity Optimal Tableaux for CPDL

Soundness, Completeness and Complexity: Let $\phi \in \mathrm{Fml}$ be a formula in negation normal form of size *n*. The procedure `is-sat`(ϕ) terminates, runs in EXPTIME in *n*, and ϕ is satisfiable iff `is-sat`(ϕ) returns true.

Implementations:

`http://users.cecs.anu.edu.au/~rpg/software.html`

Note: As far as I know, this is the only complexity-optimal algorithm for CPDL that does not use a cut rule

Optimal and Cut-Free Tableaux for Propositional Dynamic Logic with Converse. IJCAR 2010

Related Work for Fixpoint Logics

Terminating Tableaux: for LTL exist (experimentally not competitive)

Resolution Methods: LTL and CTL, optimal, PDL?, converse?

BDD-based methods: LTL, CTL, PDL plus converse, optimal

MLSolver: general solver for many modal logics

- creates and solves parity game optimally
- handles many fixpoint logics including LTL, CTL, PDL, CTL* and mu-calculus converse?
- experimentally not competitive

CTL*: Mark Renolds has developed a suboptimal tableau calculus

Mona: embed into S1S ... non-elementary complexity

PDL: automata based methods exist (converse?)

CPDL: De Giacomo and Massacci (Inf. and Comp. 2000)

- paper does not give explicit EXPTIME algorithms
- only known implementation for CPDL is unsound
- new implementation for PDL exists

Further Work

Optimisations: needed since many standard ones are not sound

Extend: to other fixpoint logics e.g. CTL, ATL, LCK

SAT: many people investigating using modern SAT/SMT solvers

QBF: worth investigating instead of SAT solvers

Efficiency and Scalability: is the front line

Questions?