Automated Reasoning in Modal, Tense and Temporal Logics

© Rajeev Goré
Logic and Computation Group
School of Computer Sciences
Australian National University
http://cecs.anu.edu.au/~rpg
Rajeev.Gore@anu.edu.au
March 7, 2011 Version 1

Contents

Lecture 1: Modal logic and modal tableaux

Syntax: atomic formulae, connectives and formulae

Atomic Formulae: $p_0, p_1, p_2, p_3, \cdots$ meta-variables p, q, r

Connectives: $\neg$, $\wedge$, $\vee$, $\rightarrow$, $[]$, $\langle\rangle$

Formulae: every atomic formula p is a formula

Formulae: if φ and ψ are formulae then so are each of $\varphi \wedge \psi$, $\varphi \vee \psi$, $\varphi \rightarrow \psi$, $[]\varphi$, $\langle\rangle\varphi$ meta-variables φ, ψ

Example: $[](p_1 \rightarrow p_2) \rightarrow ([]p_1 \rightarrow []p_1)$

Example: $[]p_4 \rightarrow [][]p_2$

Example: $\langle\rangle(p_2 \wedge p_3) \rightarrow (\langle\rangle p_2 \wedge \langle\rangle p_3)$

Semantics: worlds, accessibility relation, valuation

Kripke frame: is a pair $\langle W, R \rangle$ where

W is a non-empty set (of point/worlds/states)

$R \subseteq W \times W$ is a binary (accessibility) relation over W

Kripke model: is a triple $\langle W, R, \vartheta \rangle$ where

$\langle W, R \rangle$ is a Kripke frame

$\vartheta : W \times \mathit{Atoms} \mapsto \{\mathbf{t}, \mathbf{f}\}$ is a valuation mapping each world w and each atomic formula p to $\mathbf{t}$ or else to $\mathbf{f}$

Forcing: between worlds and formulae

$w \Vdash p$ if $\vartheta(w, p) = \mathbf{t}$ — p is true at w

$w \Vdash \neg\varphi$ if $\vartheta(w, \varphi) = \mathbf{f}$ — φ is false at w

$w \Vdash \varphi \wedge \psi$ if $\vartheta(w, \varphi) = \mathbf{t}$ and $\vartheta(w, \psi) = \mathbf{t}$

$w \Vdash \varphi \vee \psi$ if $\vartheta(w, \varphi) = \mathbf{t}$ or $\vartheta(w, \psi) = \mathbf{t}$

$w \Vdash \varphi \rightarrow \psi$ if $\vartheta(w, \varphi) = \mathbf{f}$ or $\vartheta(w, \psi) = \mathbf{t}$

$w \Vdash \langle\rangle\varphi$ if $\exists v \in W.R(w, v)$ & $\vartheta(v, \varphi) = \mathbf{t}$

i.e. some R-successor makes φ true

$w \Vdash []\varphi$ if $\forall v \in W.R(w, v) \Rightarrow \vartheta(v, \varphi) = \mathbf{t}$

i.e. every R-successor makes φ true

Example

W is the set of all students in this room

$R(w, v)$ holds if $v \in W$ is one row in front of $w \in W$

$\vartheta(w, p_1) = \mathbf{t}$ if student w has a brother

$\vartheta(w, p_2) = \mathbf{t}$ if student w has a sister

Semantics of Logical Consequence $\Gamma \models \varphi$

M is a Kripke model $\langle W, R, \vartheta \rangle$

Γ is a finite set of formulae

φ is a given formula

$M \Vdash \varphi$ if $\forall w \in W . w \Vdash \varphi$ φ is true everywhere in M

$M \Vdash \Gamma$ if $\forall \psi \in \Gamma . M \Vdash \psi$ every ψ in Γ is true everywhere in M

$\Gamma \models \varphi$ if $\forall M = \langle W, R, \vartheta \rangle . M \Vdash \Gamma \Rightarrow M \Vdash \varphi$

if Γ is true everywhere in M then φ is true everywhere in M

φ is valid: if $\emptyset \models \varphi$ φ is true everywhere in all models

φ is satisfiable: if φ is true in some world in some model

Lemma: φ is valid iff $\neg\varphi$ is not satisfiable

φ is satisfiable wrt Γ: if φ is true in some world in some model that forces Γ

Lemma: $\Gamma \models \varphi$ iff $\neg\varphi$ is not satisfiable wrt Γ

Negation Normal Form

nnf: a formula φ is in *negation normal form* if the symbol $\neg$ appears only directly before atomic formulae

Lemma: For every φ, there exists a formula $\mathit{nnf}(\varphi)$ in negation normal form such that the length of $\mathit{nnf}(\varphi)$ is only polynomially longer than that of φ, and $\varphi \leftrightarrow \mathit{nnf}(\varphi)$ is valid

Proof: Repeatedly distribute negation over subformulae using the following valid principles:

$$\models \neg\neg\varphi \leftrightarrow \varphi$$
$$\models (\varphi_1 \to \psi_1) \leftrightarrow (\neg\varphi_1 \vee \psi_1) \qquad \models \neg(\varphi_1 \to \psi_1) \leftrightarrow (\varphi_1 \wedge \neg\psi_1)$$
$$\models \neg(\varphi \wedge \psi) \leftrightarrow (\neg\varphi \vee \neg\psi) \qquad \models \neg(\varphi \vee \psi) \leftrightarrow (\neg\varphi \wedge \neg\psi)$$
$$\models \neg\langle\rangle\varphi \leftrightarrow []\neg\varphi \qquad \models \neg[]\varphi \leftrightarrow \langle\rangle\neg\varphi$$

Beware: if $\leftrightarrow$ is a primitive connective then this blows up!

Examples of negation normal form

$\models \neg\neg\varphi \leftrightarrow \varphi$

$\models (\varphi_1 \to \psi_1) \leftrightarrow (\neg\varphi_1 \vee \psi_1)$

$\models \neg(\varphi \wedge \psi) \leftrightarrow (\neg\varphi \vee \neg\psi)$

$\models \neg\langle\rangle\varphi \leftrightarrow []\neg\varphi$

$\models \neg(\varphi_1 \to \psi_1) \leftrightarrow (\varphi_1 \wedge \neg\psi_1)$

$\models \neg(\varphi \vee \psi) \leftrightarrow (\neg\varphi \wedge \neg\psi)$

$\models \neg[]\varphi \leftrightarrow \langle\rangle\neg\varphi$

Example:

$\neg([](p_0 \to p_1) \to ([]p_0 \to []p_1))$

$[](p_0 \to p_1) \wedge \neg([]p_0 \to []p_1)$

$[](p_0 \to p_1) \wedge ([]p_0 \wedge \neg[]p_1)$

$[](\neg p_0 \vee p_1) \wedge ([]p_0 \wedge \langle\rangle\neg p_1)$

Example:

$\neg([]p_0 \to p_0)$

$([]p_0) \wedge (\neg p_0)$

$\neg([]p_0 \to [][]p_0)$

$([]p_0) \wedge (\neg[][]p_0)$

$([]p_0) \wedge (\langle\rangle\neg[]p_0)$

$([]p_0) \wedge (\langle\rangle\langle\rangle\neg p_0)$

Modal Tableaux as Or-trees

Γ 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 $\{\varphi\} \cup X$
Z is saturated: if it contains no top level $\wedge, \vee, []$ formulae

Rules

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

$$(\text{K})\frac{\langle\rangle\varphi; []X; Z}{\Gamma; \varphi; X}\ Z \text{ is saturated}$$

A K-tableau for Y given global assumptions Γ

is an inverted (or) tree of nodes with:

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

A **K**-tableau is closed if all leaves are (id), else it is open.

Examples of **K**-Tableau With $\Gamma = \emptyset$

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

$$(\mathrm{K})\frac{\langle\rangle\varphi; []X; Z}{\Gamma; \varphi; X}\ Z \text{ is saturated}$$

There is a closed **K**-tableau for $\neg([](p_0 \to p_1) \to ([]p_0 \to []p_1))$

Examples of **K**-Tableau With $\Gamma = \emptyset$

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

$$(\mathrm{K})\frac{\langle\rangle\varphi; []X; Z}{\Gamma; \varphi; X}\ Z \text{ is saturated}$$

There is no closed **K**-tableau for $\neg([]p_0 \to p_0)$
There is no closed **K**-tableau for $\neg([]p_0 \to [][]p_0)$
How can we be sure, we only looked at one **K**-tableau for each ?

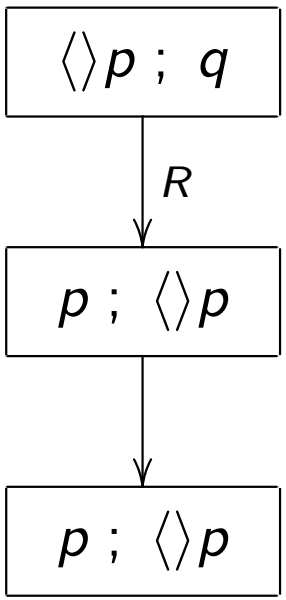
Examples

$$(\mathrm{K})\frac{\langle\rangle\varphi; []X; Z}{\Gamma; \varphi; X}$$ Z is saturated

How many different **K**-tableaux for $\langle\rangle p_1; \langle\rangle p_2; \langle\rangle \neg p_1; []p_1; []\neg p_3$?

Loops!

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



Solution: check whether new node exists already on the current branch

Soundness

$$(\mathrm{id})\frac{p;\neg p;X}{} \quad (\wedge)\frac{\varphi\wedge\psi;X}{\varphi;\psi;X} \quad (\vee)\frac{\varphi\vee\psi;X}{\varphi;X \mid \psi;X} \quad (\mathrm{K})\frac{\langle\rangle\varphi;[]X;Z}{\Gamma;\varphi;X}$$

Z is saturated

Theorem: If there is a closed **K**-tableau for $\Gamma\cup\{\neg\varphi_0\}$ then $\Gamma\models\varphi_0$.
Proof: For each rule we prove that if the premiss is **K**-satisfiable then so is at least one conclusion.

- (id): if $p;\neg p;X$ is satisfiable then ...
- $(\vee)$: if $X;\varphi\vee\psi$ is **K**-satisfiable then so is $X;\varphi$ or $X;\psi$
- $(\wedge)$: if $X;\varphi\wedge\psi$ is **K**-satisfiable then so is $X;\varphi;\psi$
- (K): if $\langle\rangle\varphi;[]X;Z$ is **K**-satisfiable then so is $\varphi;X$

Each branch $n_0, n_1, \cdots, n_k$ of nodes has $n_0=\Gamma\cup\{\neg\varphi_0\}$ and $n_k=\{p,\neg p\}\cup X$ for some set X and some atomic formula p.
So, if n_0 is **K**-satisfiable then n_1 is **K**-satisfiable ... then $p;\neg p;X$ is **K**-satisfiable. Contradiction. This applies to every branch.
So $\Gamma\cup\{\neg\varphi_0\}$ is not **K**-satisfiable i.e. $\forall M.M\Vdash\Gamma\Rightarrow M\Vdash\varphi_0$

Completeness

Theorem: If $\Gamma \models \varphi_0$ then there is a closed **K**-tableau for $\Gamma \cup \{\neg\varphi_0\}$.

Proof: We prove the contra-positive:

if there is no closed **K**-tableau for $\Gamma \cup \{\neg\varphi_0\}$ then $\Gamma \not\models \varphi_0$.

Assume: that every **K**-tableau for $\Gamma \cup \{\neg\varphi_0\}$ is open

Show: that $\Gamma \cup \{\neg\varphi_0\}$ is **K**-satisfiable i.e.
$\exists M = \langle W, R, \vartheta \rangle . \exists w \in W . M \Vdash \Gamma \& w \Vdash \neg\varphi_0$

Complexity and Optimisations

2EXPTIME: we can explore the same node on multiple branches

Optimisations: practical implementations use many optimisations

Lecture 2: Description Logic with Inverse Roles

Syntax: concepts and roles

Concept Names: $A, B ::= a_0 \mid a_1 \mid a_2 \mid \cdots$

Role Names: $R, S ::= r_0 \mid r_1 \mid r_2 \mid \cdots$

Concepts: $C, D ::= \top \mid \bot \mid A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \forall R.C \mid \exists R.C$

TBox: finite set of "axioms" of the form $C \sqsubseteq D$ or $C = D$.

NNF: later assume that all formulae are in Negation Normal Form

Semantics of Description Logics

Interpretation: $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ consists of a non-empty (domain) set $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ that maps every concept name A to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ and maps every role name R to a binary relation $R^{\mathcal{I}}$ on $\Delta^{\mathcal{I}}$

Interpretation: of complex concepts is as follows

$$\begin{array}{lcl} \top^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \\ \bot^{\mathcal{I}} & = & \emptyset \\ (\neg C)^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\ (C \sqcap D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cap D^{\mathcal{I}} \\ (C \sqcup D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cup D^{\mathcal{I}} \end{array}$$

Intuition:: we have classical propositional logic at least

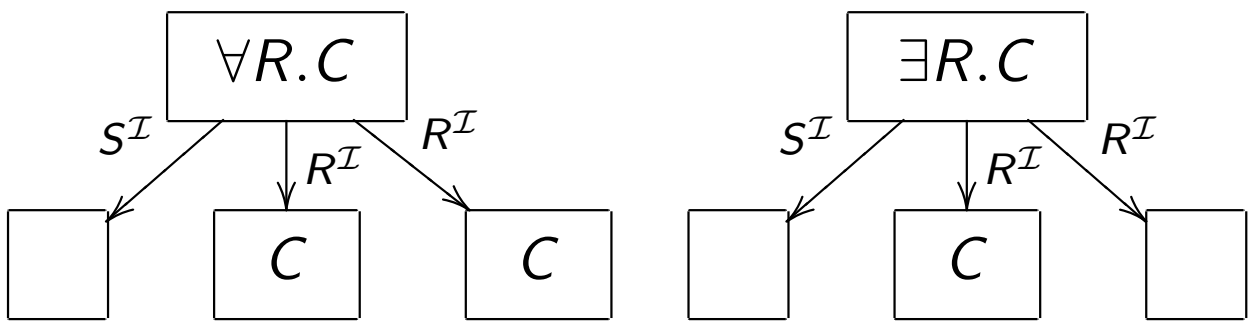
Semantics of Description Logics

Interpretation: $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ consists of a non-empty (domain) set $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ that maps every concept name A to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ and maps every role name R to a binary relation $R^{\mathcal{I}}$ on $\Delta^{\mathcal{I}}$

Interpretation: of complex concepts is as follows

$$(\forall R.C)^{\mathcal{I}} = \left\{ x \in \Delta^{\mathcal{I}} \mid \forall y \left[(x, y) \in R^{\mathcal{I}} \text{ implies } y \in C^{\mathcal{I}} \right] \right\}$$

$$(\exists R.C)^{\mathcal{I}} = \left\{ x \in \Delta^{\mathcal{I}} \mid \exists y \left[(x, y) \in R^{\mathcal{I}} \text{ and } \ y \in C^{\mathcal{I}} \right] \right\}$$



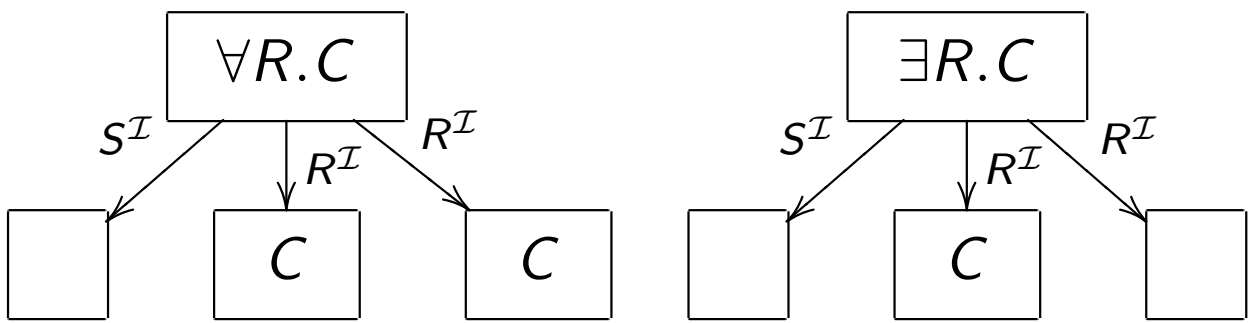
Semantics of Description Logics

Interpretation: $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ consists of a non-empty (domain) set $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ that maps every concept name A to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ and maps every role name R to a binary relation $R^{\mathcal{I}}$ on $\Delta^{\mathcal{I}}$

Interpretation: of complex concepts is as follows

$\forall R.C$ means every $R^{\mathcal{I}}$-successor makes C true

$\exists R.C$ means some $R^{\mathcal{I}}$-successor makes C true



Semantics of Description Logics

Interpretation: $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ consists of a non-empty (domain) set $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ that maps every concept name A to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ and maps every role name R to a binary relation $R^{\mathcal{I}}$ on $\Delta^{\mathcal{I}}$

$\mathcal{I}$ *satisfies* C: if $C^{\mathcal{I}} \neq \emptyset$ (true somewhere under $\mathcal{I}$)

$\mathcal{I}$ validates C: if $C^{\mathcal{I}} = \Delta^{\mathcal{I}}$ (true everywhere under $\mathcal{I}$)

Prop: $\mathcal{I}$ *validates* C iff it does not *satisfy* $\neg C$

$\mathcal{I}$ is a model of TBox $\mathcal{T}$: if for every axiom $C \sqsubseteq D$ (resp. $C = D$) of $\mathcal{T}$, we have $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ (resp. $C^{\mathcal{I}} = D^{\mathcal{I}}$)

$\mathcal{I}$ satisfies a set X of concepts if there exists $x \in \Delta^{\mathcal{I}}$ such that $x \in C^{\mathcal{I}}$ for all $C \in X$

X is satisfiable w.r.t. TBox $\mathcal{T}$: if there exists a model of $\mathcal{T}$ that satisfies X.

Negation Normal Form

nnf: a concept C is in *negation normal form* if the symbol $\neg$ appears only directly before atomic concepts

Lemma: For every C, there exists a concept $nnf(C)$ in negation normal form such that the length of $nnf(C)$ is only polynomially longer than that of C, and $C \leftrightarrow nnf(C)$ is valid

Proof: Repeatedly distribute negation over subconcepts using the following valid principles:

$$\begin{array}{rcc@{\qquad}rcc}
\neg\neg C & = & C & & & \\
(C \sqsubseteq D) & = & (\neg C \sqcup D) & \neg(C \sqsubseteq D) & = & (C \sqcap \neg D) \\
\neg(C \sqcap D) & = & (\neg C \sqcup \neg D) & \neg(C \sqcup D) & = & (\neg C \sqcap \neg D) \\
\neg\exists R.C & = & \forall R.\neg C & \neg\forall R.C & = & \exists R.\neg C
\end{array}$$

Beware: if $=$ is a primitive connective then this blows up!

Description Logic Tableaux Are And-Trees

$$(\text{id})\frac{X\ ;\ A\ ;\ \neg A}{} \qquad (\sqcap)\frac{X\ ;\ C_1 \sqcap C_2}{X\ ;\ C_1\ ;\ C_2} \qquad (\sqcup)\frac{X\ ;\ C_1 \sqcup C_2}{X\ ;\ C_i}\ i \in \{1,2\}$$

$$(\exists)\frac{X\ ;\ \exists R_1.C_1\ ;\ \cdots\ ;\ \exists R_n.C_n}{C_1\ ;\ X_1\ ||\ \cdots\ ||\ C_n\ ;\ X_n}\ X_i = \{D : \forall R_i.D \in X\}$$

$(\sqcup)$ rule does not cause any explicit branching in a tableau

$(\exists)$ creates an R_i-child node for each $\exists R_i.C_i$, $1 \leq i \leq n$

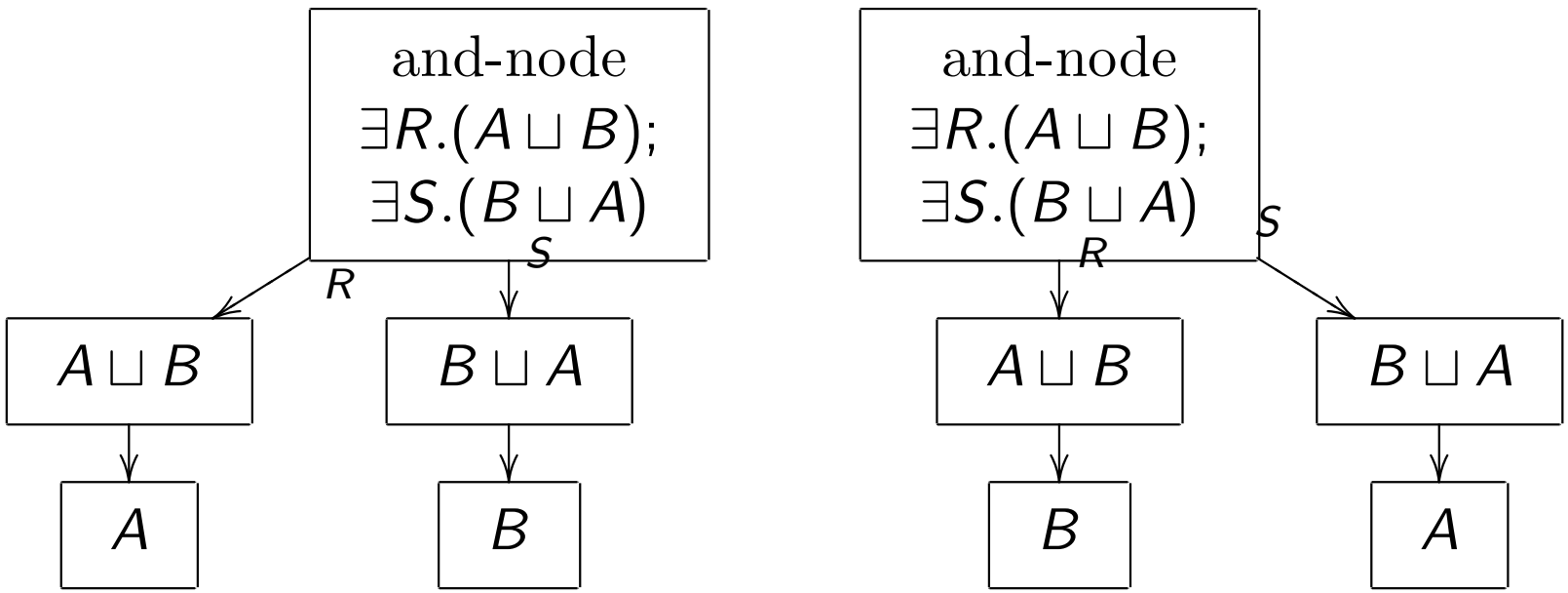
$||$ flags and-branching i.e. parent open if all children are open

Models: satisfiability requires only one open child for $\sqcup$-nodes but requires an open child for each $\exists R_i.C_i$

Description Logic Tableaux Are And-Trees

$$(\text{id})\frac{X\ ;\ A\ ;\ \neg A}{} \qquad (\sqcap)\frac{X\ ;\ C_1 \sqcap C_2}{X\ ;\ C_1\ ;\ C_2} \qquad (\sqcup)\frac{X\ ;\ C_1 \sqcup C_2}{X\ ;\ C_i}\ i \in \{1,2\}$$

$$(\exists)\frac{X\ ;\ \exists R_1.C_1\ ;\ \cdots\ ;\ \exists R_n.C_n}{C_1\ ;\ X_1 \,||\, \cdots \,||\, C_n\ ;\ X_n}\ X_i = \{D : \forall R_i.D \in X\}$$



All nodes can be considered to be and-nodes i.e. open if all children open

Handling a non-empty TBox $\mathcal{T}$

Change the (E) rule:

$$(\exists)\frac{X\ ;\ \exists R_1.C_1\ ;\ \cdots\ ;\ \exists R_n.C_n}{C_1\ ;\ X_1\ ||\ \cdots\ ||\ C_n\ ;\ X_n}\ X_i = \{D : \forall R_i.D \in X\}$$

$$(\exists)\frac{X\ ;\ \exists R_1.C_1\ ;\ \cdots\ ;\ \exists R_n.C_n}{nnf(\mathcal{T})\ ;\ C_1\ ;\ X_1\ ||\ \cdots\ ||\ nnf(\mathcal{T})\ ;\ C_n\ ;\ X_n}\ X_i = \{D : \forall R_i.D \in X\}$$

Blocking: branch can cycle back to an ancestor (loop check)

Thm: X is satisfiable w.r.t. TBox $\mathcal{T}$ iff there is an open and-tree tableau for the root node containing $\mathcal{T}\ ;\ X$

Note: real DL provers do not use this naive technique to handle TBoxes but use more sophisticated techniques (model merging)

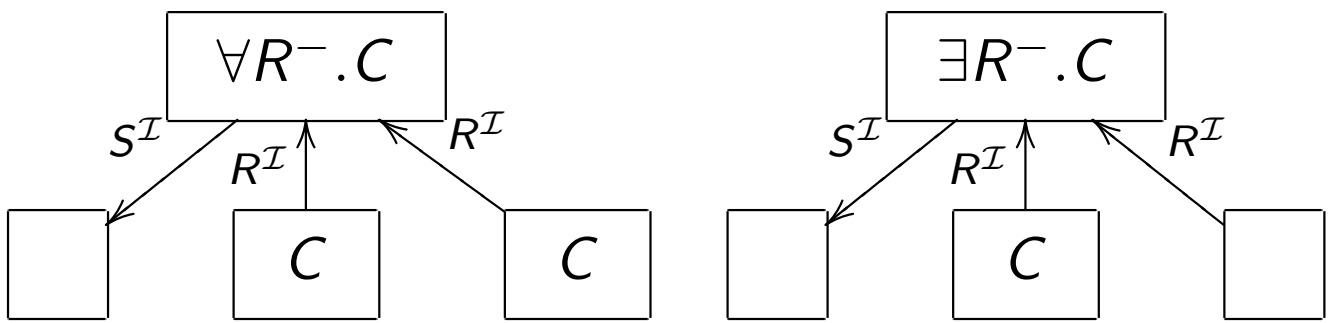
Inverse Roles

Syntax: allow $\forall R^{-}.C$ and $\exists R^{-}.C$

Interpretation: of complex concepts is as follows

$$(\forall R^{-}.C)^{\mathcal{I}} = \left\{x \in \Delta^{\mathcal{I}} \mid \forall y \left[(y,x) \in R^{\mathcal{I}} \text{ implies } y \in C^{\mathcal{I}}\right]\right\}$$

$$(\exists R^{-}.C)^{\mathcal{I}} = \left\{x \in \Delta^{\mathcal{I}} \mid \exists y \left[(y,x) \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\right]\right\}$$



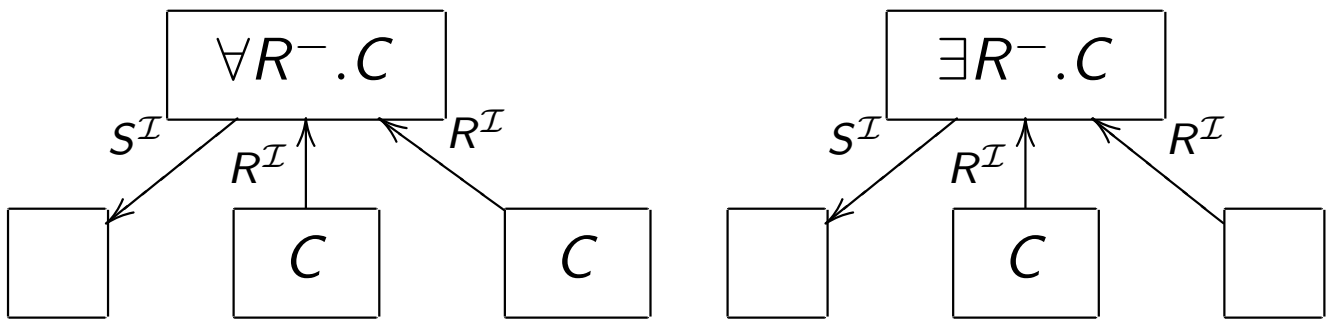
Inverse Roles

Syntax: allow $\forall R^-.C$ and $\exists R^-.C$

Interpretation: of complex concepts is as follows

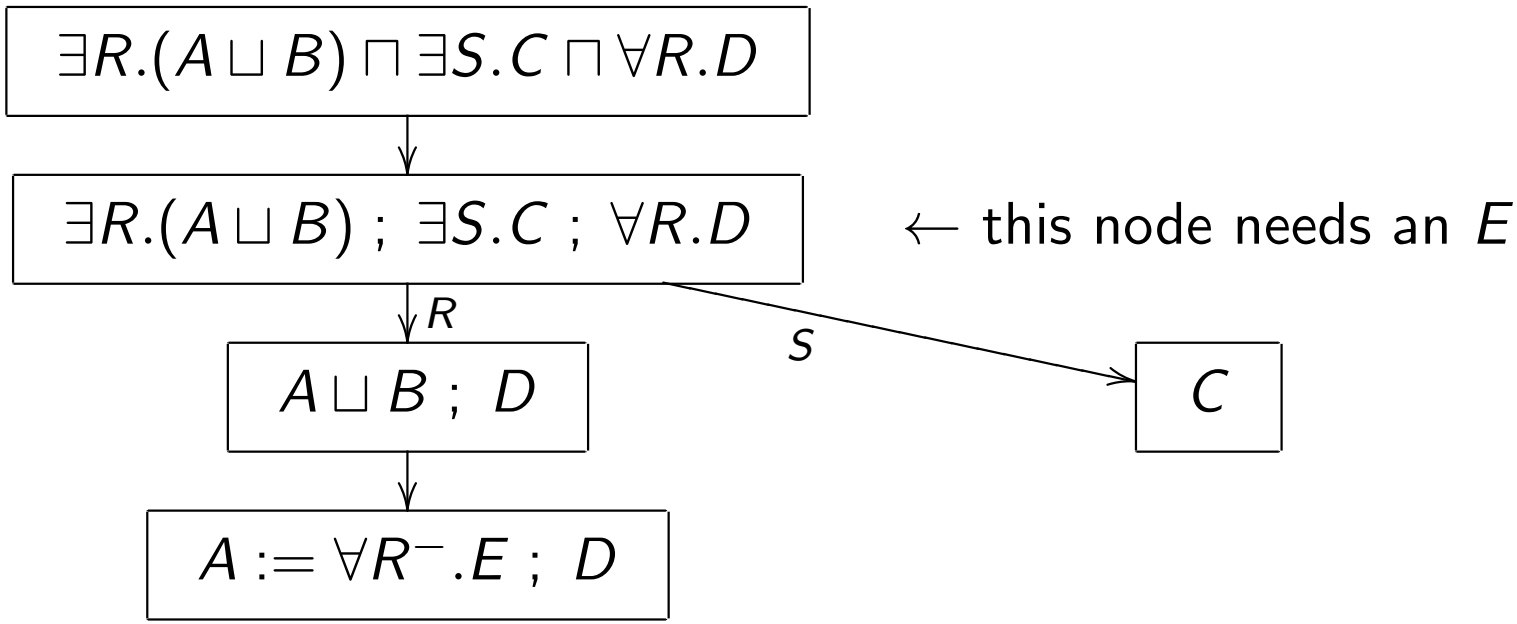
$(\forall R^-.C)^{\mathcal{I}}$ means every $R^{\mathcal{I}}$-predecessor makes C true

$(\exists R^-.C)^{\mathcal{I}}$ means some $R^{\mathcal{I}}$-predecessor makes C true



Intuition: inverse roles "look" against arrows

Inverse Roles Can Cause Incompatibility



R-parent: (state) is missing the required concept E

Solution: backtrack to parent and "restart" by adding E

Dynamic Blocking: all nodes blocked on old state need to be unblocked after adding E e.g. some descendant of C

Suboptimal: if this A-branch closes, same work may be required on B-branch and we have to repeat this work

Summary

Description logic ALC: is just multimodal logic K_n where:

$\forall R.C$	is analogue of	$[R]\varphi$	all R-successors
$\exists R.C$	is analogue of	$\langle R\rangle\varphi$	some R-successor

Semantics: of both is in terms of multi-binary-relational frames $\langle W, R_1, R_2, \cdots, R_n\rangle$

Inverse roles: are just "converse" modalities

Description logic ALCI: is just multi-modal tense logic Kt_n where

$\forall R.C$	is analogue of	$[R]\varphi$	all R-successors
$\exists R.C$	is analogue of	$\langle R\rangle\varphi$	some R-successor
$\forall R^-.C$	is analogue of	$[R^{-1}]\varphi$	all R-predecessors
$\exists R^-.C$	is analogue of	$\langle R^{-1}\rangle\varphi$	some R-predecessor

Traditional modal tableaux: are or-trees

Traditional DL tableaux: are and-trees

Both: methods provide suboptimal solutions with worst-case 2EXPTIMEbehaviour for an EXPTIME-complete problem

Lecture 3:Complexity-optimal Tableaux Using and-or Graphs

Motivation: can we combine optimality and practicality?

Theory: ALC-satisfiability wrt a TBox is EXPTIME-complete

Practice: traditional modal- and DL- tableaux are suboptimal

Okay: if done deliberately for practical reasons

- Example: Knuth-Morris-Pratt (optimal) versus Boyer-Moore (suboptimal) for string matching
- Example: Linear (optimal) versus Robinson (suboptimal) for unification

Not Okay: if known optimal methods "not so easy to understand"

Question: Can we get the best of both worlds?

Motivation: can we combine optimality and practicality?

Theory: ALC-satisfiability wrt a TBox is EXPTIME-complete

Practice: traditional modal- and DL- tableaux are suboptimal

Okay: if done deliberately for practical reasons

- Example: Knuth-Morris-Pratt (optimal) versus Boyer-Moore (suboptimal) for string matching
- Example: Linear (optimal) versus Robinson (suboptimal) for unification

Not Okay: if known optimal methods "not so easy to understand"

Question: Can we get the best of both worlds?

Motivation: can we combine optimality and practicality?

Theory: ALC-satisfiability wrt a TBox is EXPTIME-complete

Practice: traditional modal- and DL- tableaux are suboptimal

Okay: if done deliberately for practical reasons

- Example: Knuth-Morris-Pratt (optimal) versus Boyer-Moore (suboptimal) for string matching
- Example: Linear (optimal) versus Robinson (suboptimal) for unification

Not Okay: if known optimal methods "not so easy to understand"

Question: Can we get the best of both worlds?

Modal Logic Tableaux Are Or-Trees

$$(\sqcup)\ \frac{X\ ;\ C_1 \sqcup C_2}{X\ ;\ C_1 \quad | \quad X\ ;\ C_2} \qquad (\exists)\ \frac{X\ ;\ \exists R.C}{C\ ;\ \{D : \forall R.D \in X\}}$$

$(\sqcup)$ rule creates explicit or-branching in a tableau

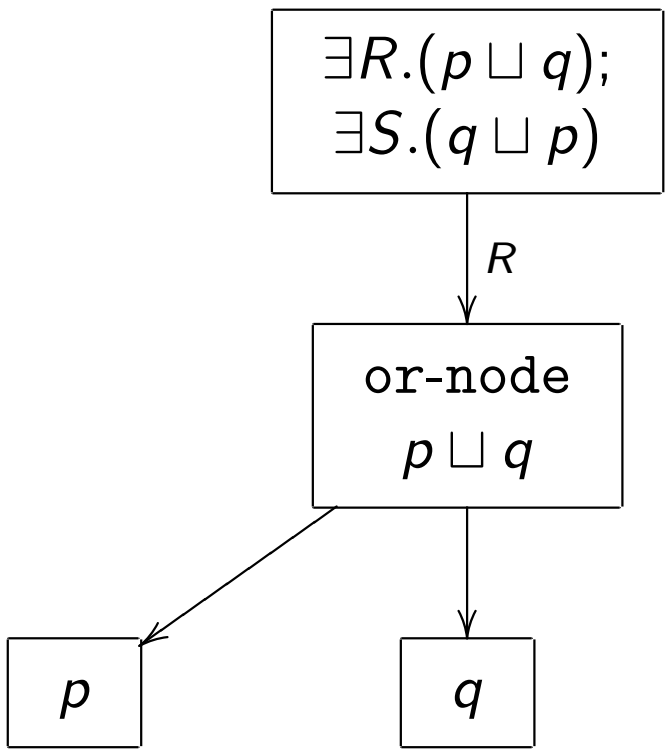
$|$ flags or-branching i.e. parent `unsat` iff both children are `unsat`

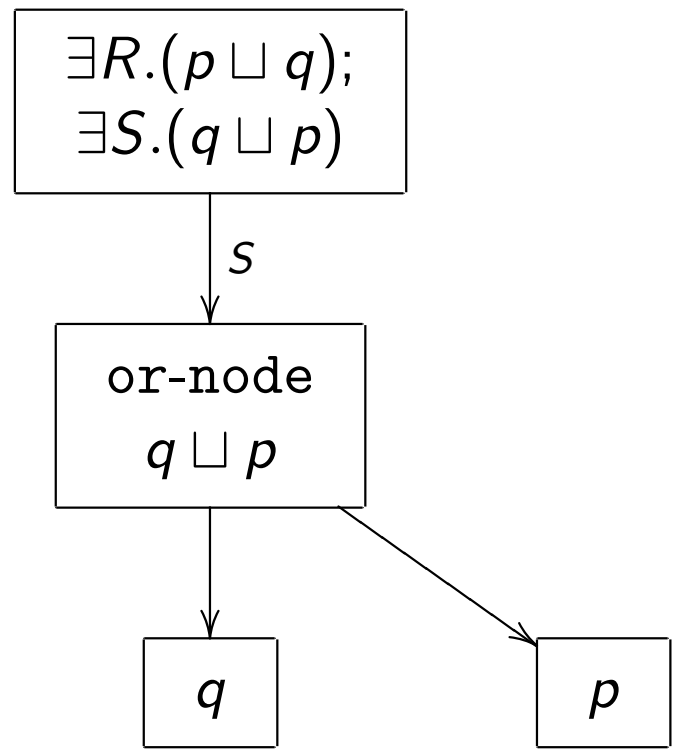
$(\exists)$ non-deterministically creates an R-child node for some $\exists R.C$

Proofs: unsatisfiability requires us to close both children of an $\sqcup$-node but requires us to close the child of only one $\exists R_i.C_i$

Modal Logic Tableaux Are Or-Trees

$$(\sqcup)\ \frac{X\ ;\ C_1 \sqcup C_2}{X\ ;\ C_1 \quad | \quad X\ ;\ C_2} \qquad (\exists)\ \frac{X\ ;\ \exists R.C}{C\ ;\ \{D : \forall R.D \in X\}}$$





Description Logic Tableaux Are And-Trees

$$(\sqcup)\ \frac{X\ ;\ C_1 \sqcup C_2}{X\ ;\ C_i}\quad i \in \{1,2\} \qquad (\exists)\ \frac{X\ ;\ \exists R_1.C_1\ ;\ \cdots\ ;\ \exists R_n.C_n}{C_1\ ;\ X_1 \mid\mid \cdots \mid\mid C_n\ ;\ X_n}$$

$$X_i = \{D : \forall R_i.D \in X\} \text{ for } 1 \leq i \leq n$$

$(\sqcup)$ rule does not cause any explicit branching in a tableau

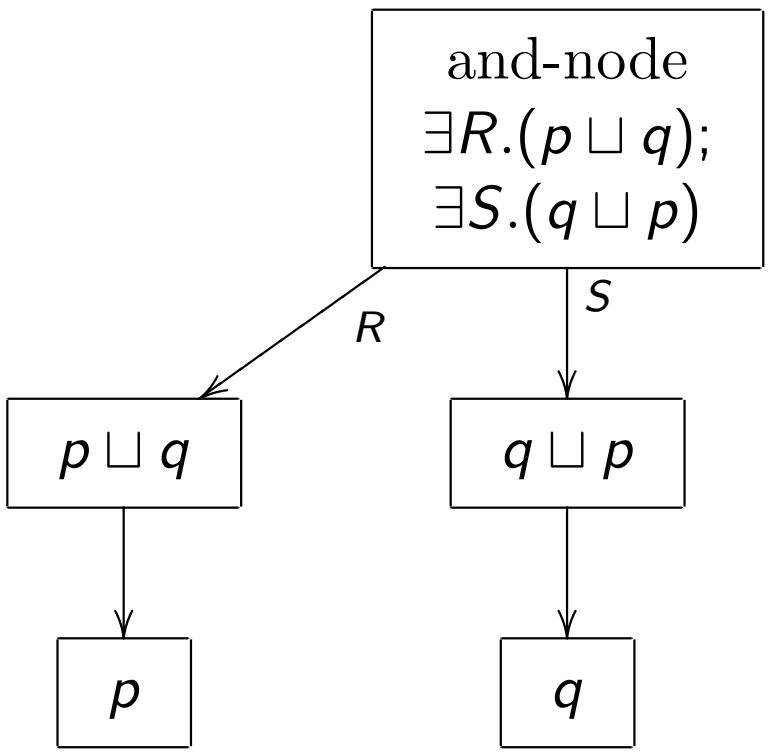
$(\exists)$ creates an R_i-child node for each $\exists R_i.C_i$, $1 \leq i \leq n$

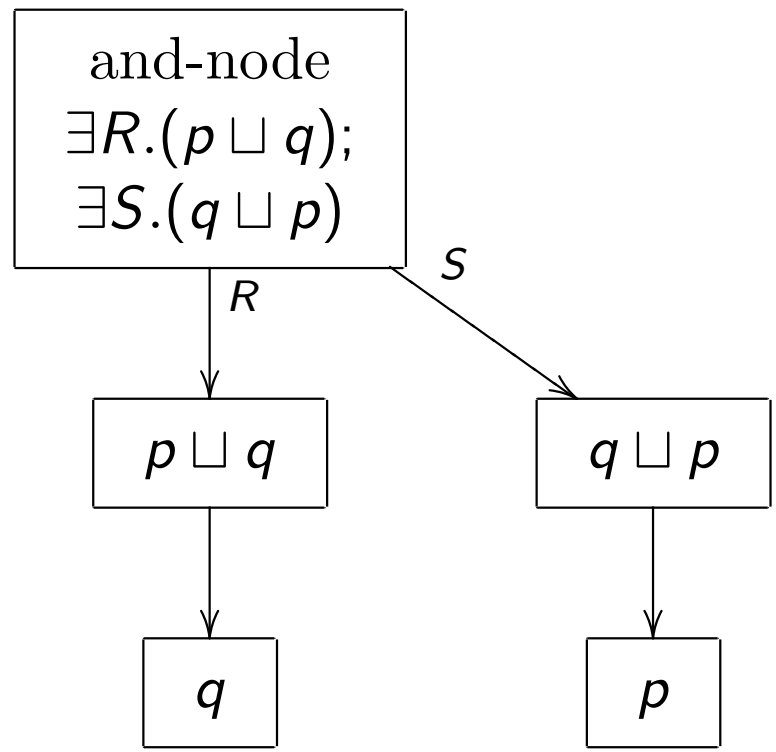
$\mid\mid$ flags and-branching i.e. parent `sat` iff all children are `sat`

Models: satisfiability requires only one open child for $\sqcup$-nodes but requires an open child for each $\exists R_i.C_i$

Description Logic Tableaux Are And-Trees

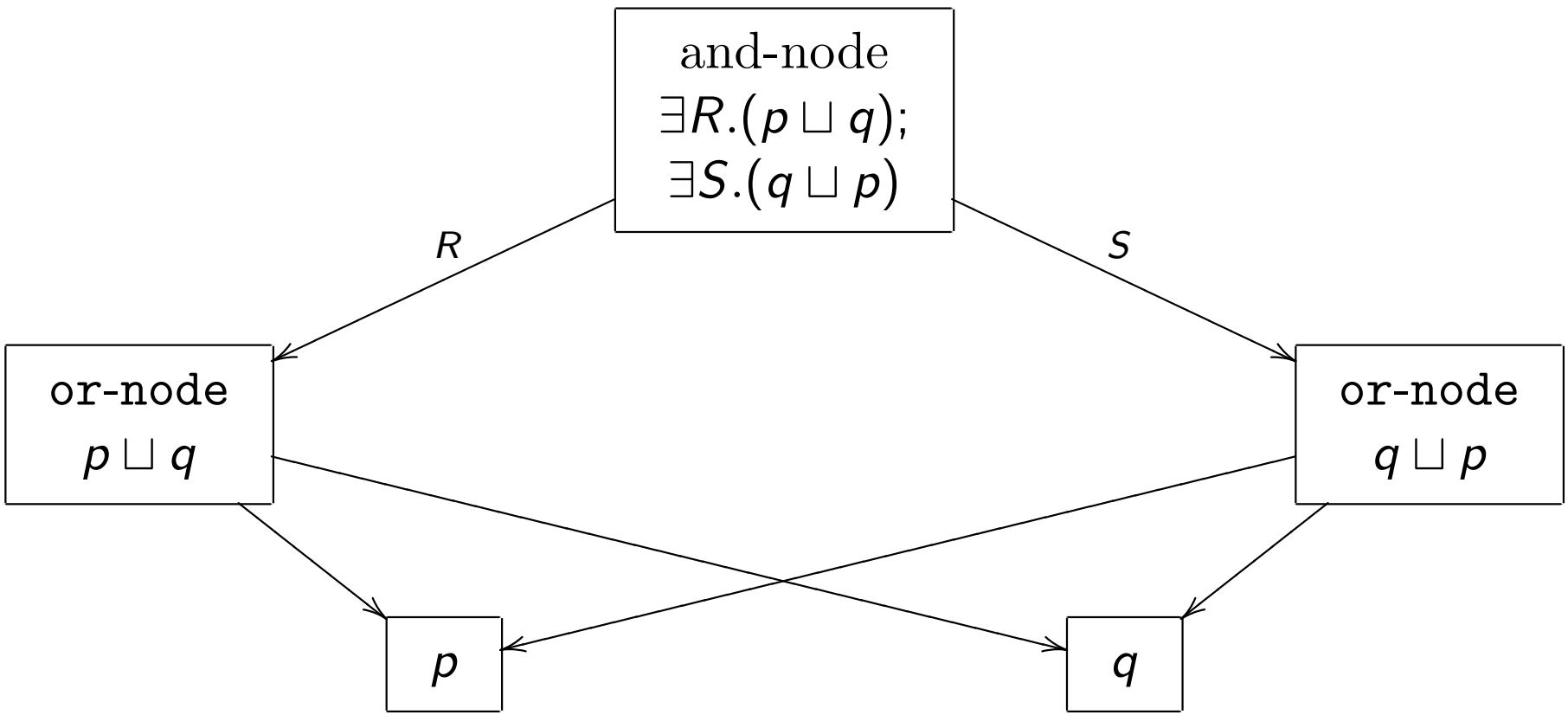
$$(\sqcup)\ \frac{X\ ;\ C_1 \sqcup C_2}{X\ ;\ C_i} \quad i \in \{1,2\} \qquad (\exists)\ \frac{X\ ;\ \exists R_1.C_1\ ;\ \cdots\ ;\ \exists R_n.C_n}{C_1\ ;\ X_1 \ ||\ \cdots\ ||\ C_n\ ;\ X_n}$$





Search Space is an And-Or Tree

$$(\sqcup)\ \frac{X\ ;\ C_1 \sqcup C_2}{X\ ;\ C_1 \mid X\ ;\ C_2} \qquad (\exists)\ \frac{X\ ;\ \exists R_1.C_1\ ;\ \cdots\ ;\ \exists R_n.C_n}{C_1\ ;\ X_1 \parallel \cdots \parallel C_n\ ;\ X_n}$$



Termination Via Blocking (aka Loop-Check)

Block a node from expansion if we have seen it before

ancestor equality blocking: within same (and- or or-) tree

ancestor subset blocking: within same (and- or or-) tree

anywhere blocking: within same and-tree

dynamic blocking: within same and-tree

pair-wise anywhere blocking: within same and-tree

Status of a blocked node is immediately classified as open even though its logical status is "unknown" rather than satisfiable.

closed $= \texttt{unsat}$ open $= \{\texttt{sat}, \texttt{blocked}\}$

Problem: traditional methods are suboptimal

TBoxes (aka global assumptions): imply complexity of deciding satisfiability is now EXPTIME-complete

Longest branch: can now contain an exponential number of nodes

Branching: means that we can explore double-exponential number of nodes in worst case

Culprit: we explore the same node on different or-branches

Anywhere blocking: does not help since it is restricted to the same and-tree

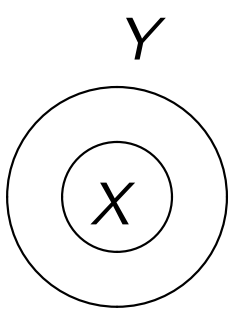
Caching: remember the status of previously seen nodes

Caching: store the `sat` or `unsat` status of previously seen nodes in a look-up table

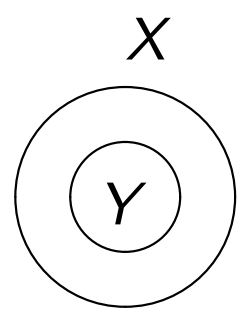
Use: when creating new tableau node, first check whether this node exists in the cache, and attach same status to new node

Assuming current node contains X: if cache contains a node $Y \supseteq X$ (respectively $Y \subseteq X$) and Y has status `sat` (`unsat`) then the current node is given the same status

$status(Y) = \texttt{sat}$



$status(Y) = \texttt{unsat}$



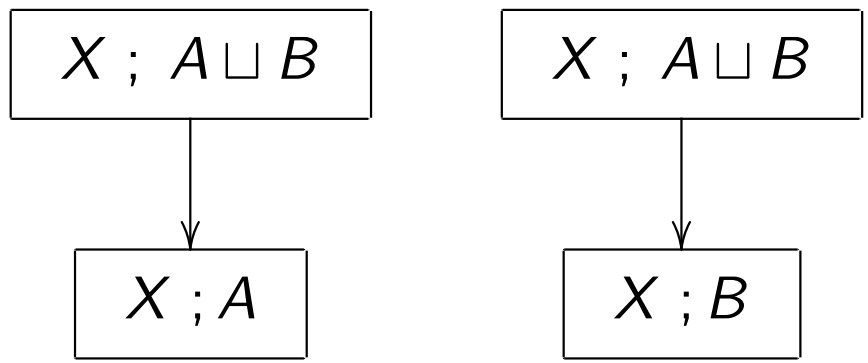
Caching Versus Blocking

Cached nodes must have a status of `sat` or `unsat`

Blocked nodes must have a status of open

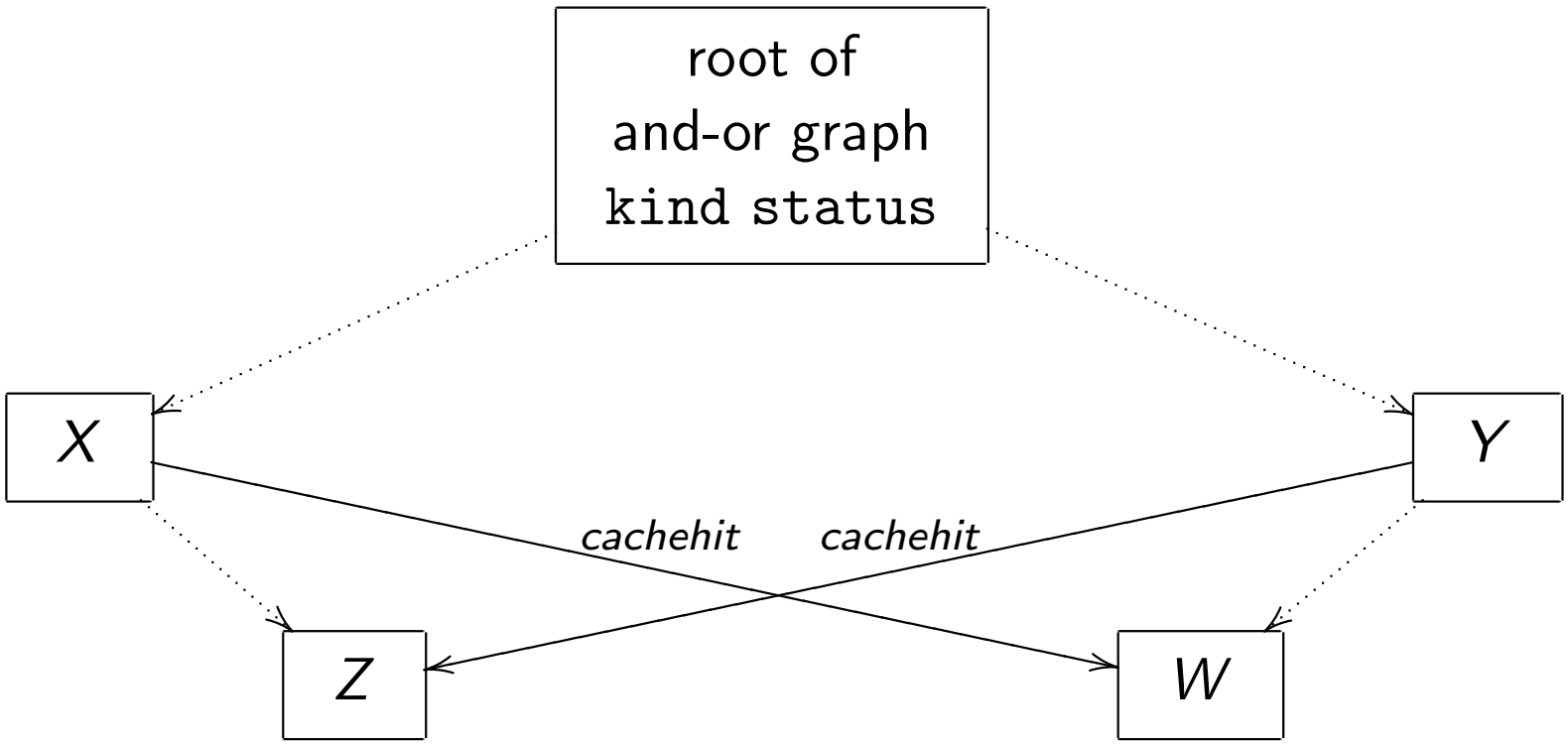
Mixed Caching

Cache blocked nodes in same and-tree as `sat`



Must remove such nodes from cache when moving to a different run

Global Caching: never explore the same node twice



Kind: rule application turns each node into `or-node` or and-node

Status: `unexpanded`, `expanded`, `unsat`, `sat`

Search strategy: immaterial for *ALC*

Global Caching Algorithm Outline

$$(\sqcup)\ \frac{X\ ;\ C_1 \sqcup C_2}{X\ ;\ C_1 \mid X\ ;\ C_2} \qquad (\exists)\ \frac{X\ ;\ \exists R_1.C_1\ ;\ \cdots\ ;\ \exists R_n.C_n}{C_1\ ;\ X_1\ ;\mathcal{T} \mid\mid \cdots \mid\mid C_n\ ;\ X_n\ ;\mathcal{T}}$$

WHILE there are unexpanded nodes, pick one say x

- if $\{A, \neg A\} \subseteq x$ then $x.\mathit{status} = \texttt{unsat}$
- ElseIf no rule applicable to x then $x.\mathit{status} = \texttt{sat}$
- Else apply $(\sqcap)$ or $(\sqcup)$ or $(\exists)$ rule to x, create children **only if they don't exist in** G, add appropriate edges, mark x as expanded
- if x.status $\in \{\texttt{unsat}, \texttt{sat}\}$ then propagate iteratively

END

Result: if root.status $= \texttt{unsat}$ then $\texttt{unsat}$ else $\texttt{sat}$

Propagation: Determining the status of nodes

`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`

Theorems

Thm: algorithm terminates

Proof: finite number of different nodes $\rightsquigarrow$ eventually all nodes exist in G $\rightsquigarrow$ each iteration expands one node

Thm: root.status $=$ `unsat` implies X is unsatisfiable wrt $\mathcal{T}$

Proof: by induction on the order in which nodes become `unsat`

Thm: root.status $=$ `sat` implies X is satisfiable wrt $\mathcal{T}$

Proof: any traditional modal- or DL- tableau has an open branch

Intuition: unwind cross-branch edges by copying nodes until ancestor-cylce found

Thm: Worst-case complexity is EXPTIME

Proof: in worst case, we explore all 2^n subsets of $Sf(\mathcal{T} \cup X)$

Implications of Global Caching

Termination: trivial

EXPTIME worst-case complexity: immediate

(Anywhere) Equality Blocking: comes for free

(Global) `unsat` caching: comes for free

Sound (Global) `sat` caching: comes for free

Individual reuse: comes for free

Experimental Results

One framework: written in C++

Implements: no-caching, unsat-caching, mixed-caching, global-caching, all with backjumping (use-check)

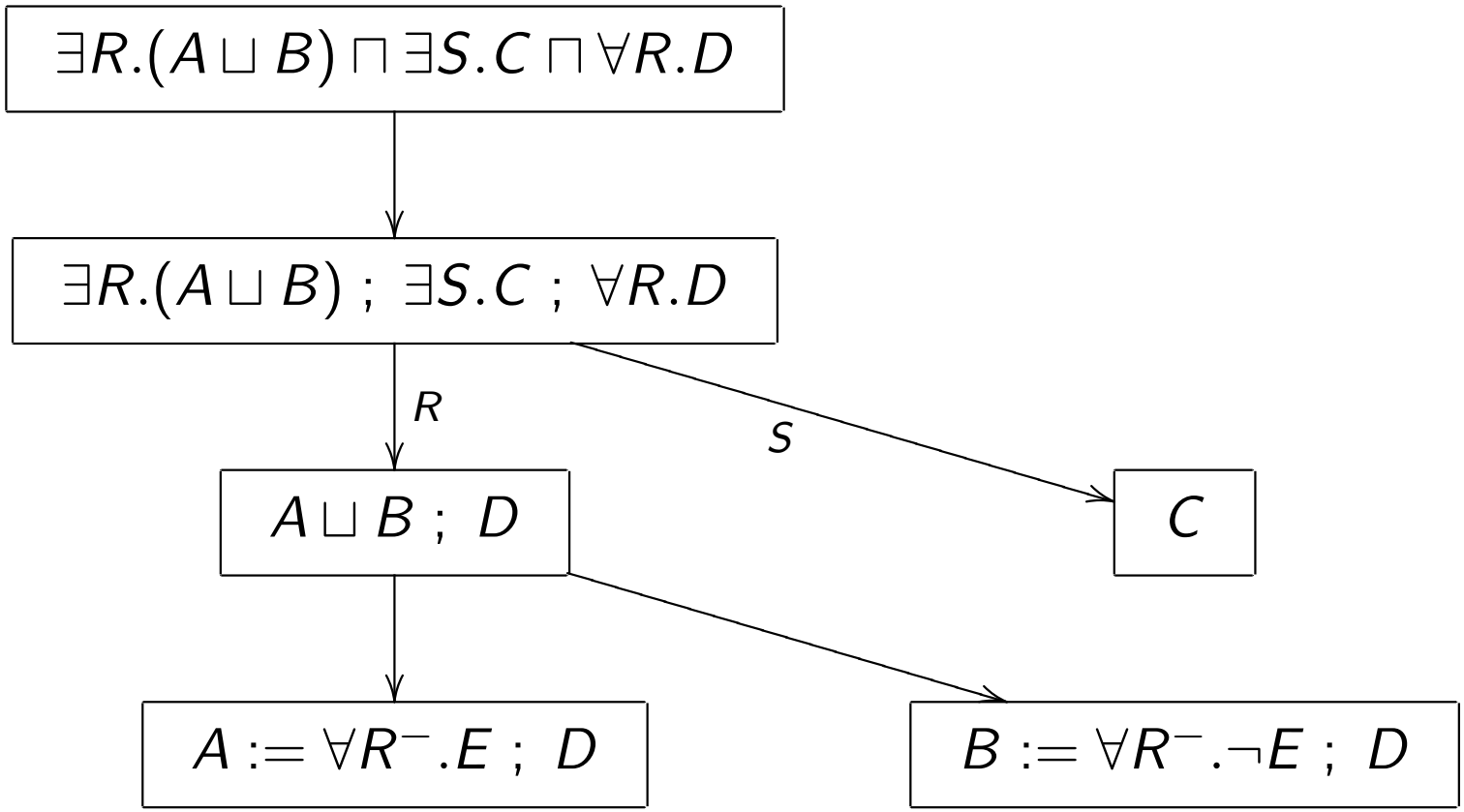
Benchmarks: TANCS 1999 modal formulae with TBoxes

Conclusions: global caching is competitive with mixed-caching

Pathological examples exist: where mixed-caching is arbitrarily worse ... reverse not possible

But: could not reproduce this behaviour in FaCT++

Inverse Roles Can Cause Incompatibility



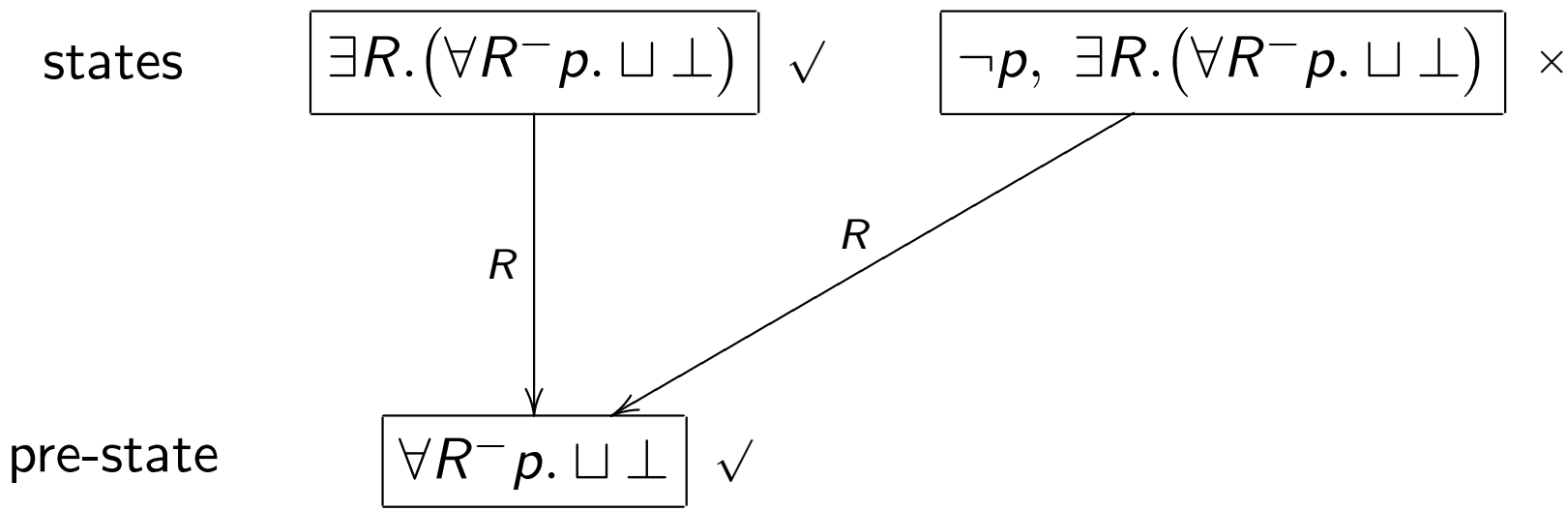
Traditional Solution for Handling Inverse Roles

- Choose disjunctions non-deterministically and add choice-point
- Add formulae to nodes as needed
- Implementation must revert all changes when backtracking
- Plus: event driven ... no unnecessary guessing
- Minus: not optimal (NEXPTIME)
- Minus: need dynamic blocking since node contents change over time

Global Caching Not Possible

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

Non-states are pre-states



Satisfiability of pre-state does not imply satisfiability of state

Our Alternative Method

Starting point: "Sound Global Caching for 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

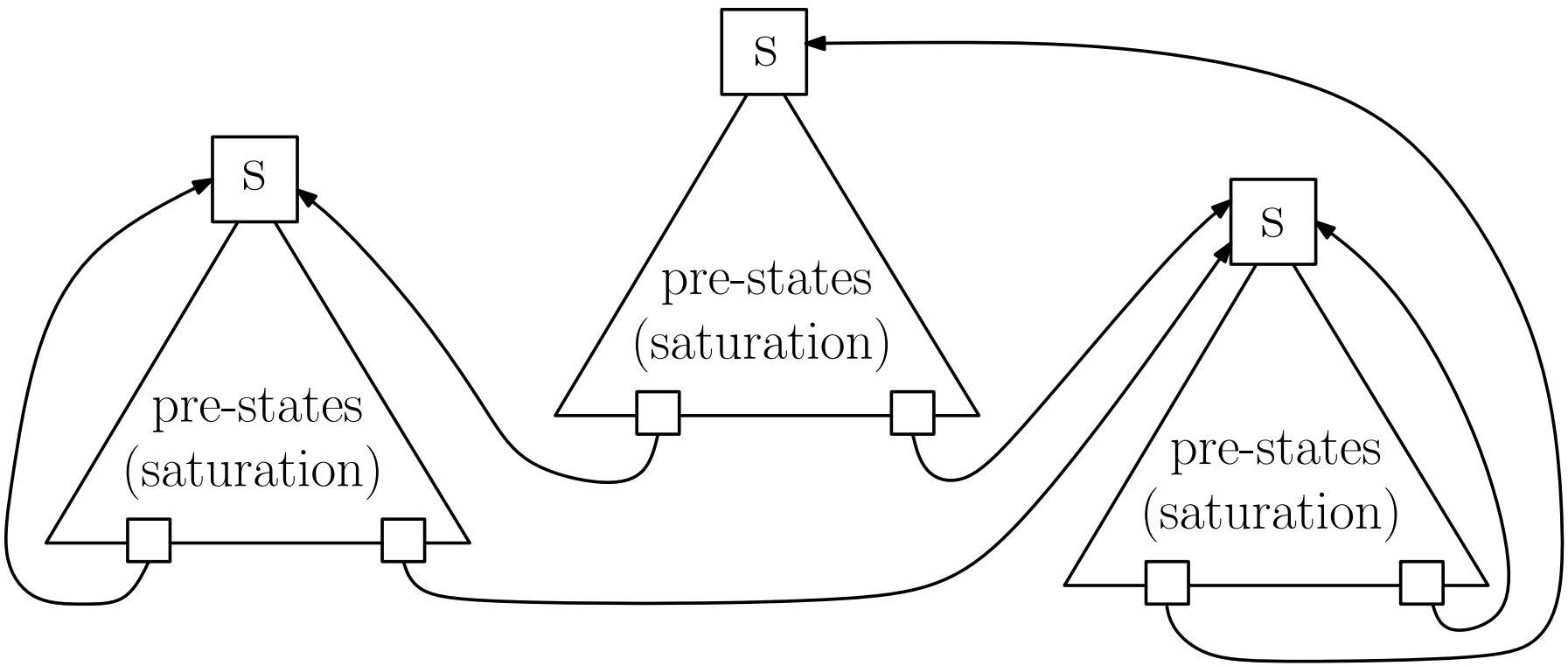
But: status of sat, unsat, or toosmall will not change

Global State Caching: never explore the same state again

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

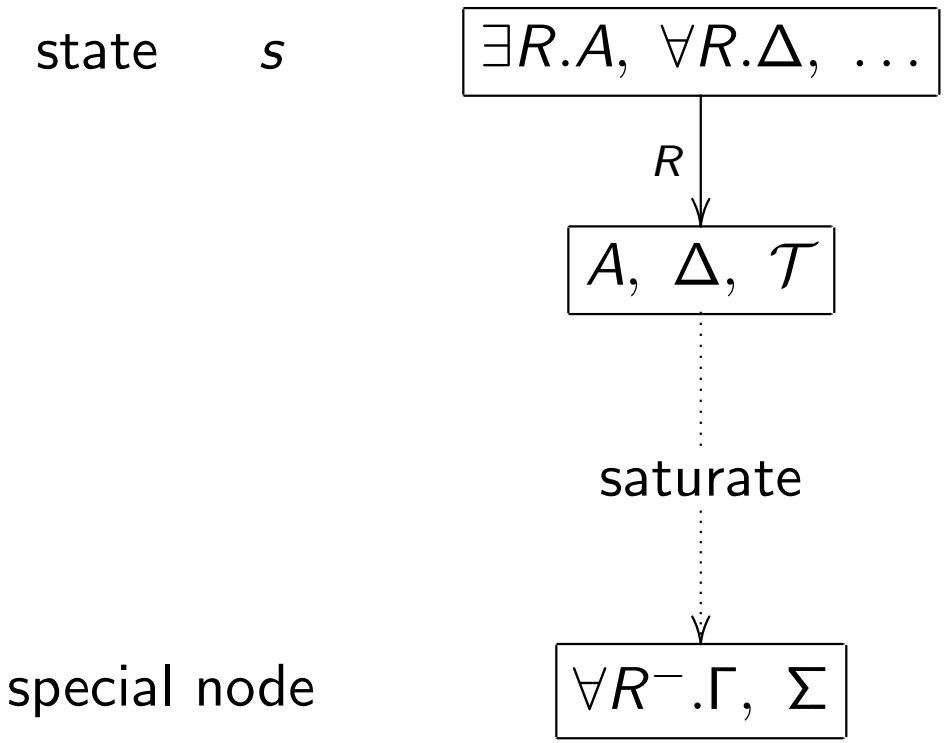
Global state caching: no two *states* with the same set of formulae

Saturation: apply $\sqcap$ and $\sqcup$ rules until not applicable (giving state)



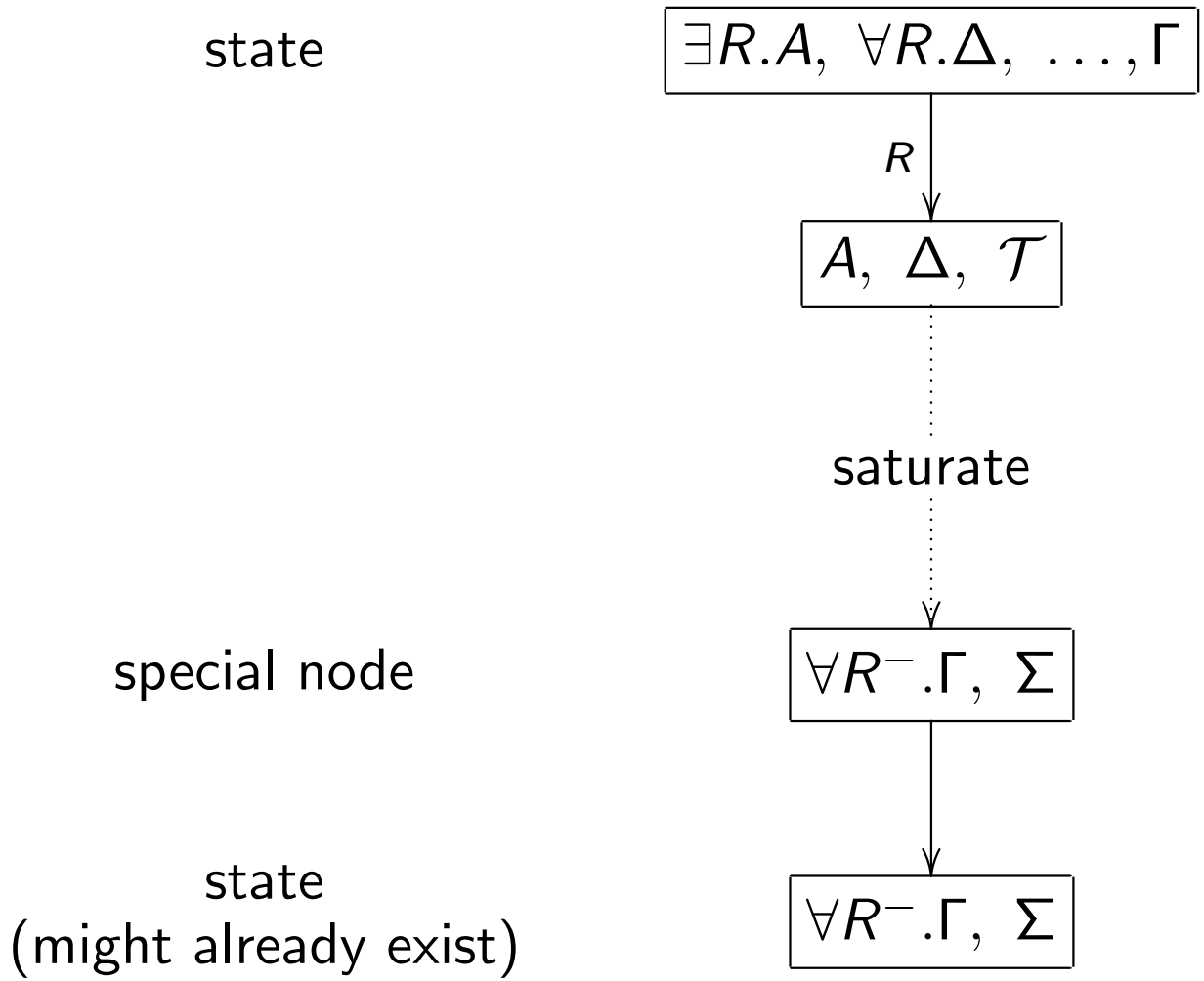
Special Nodes

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

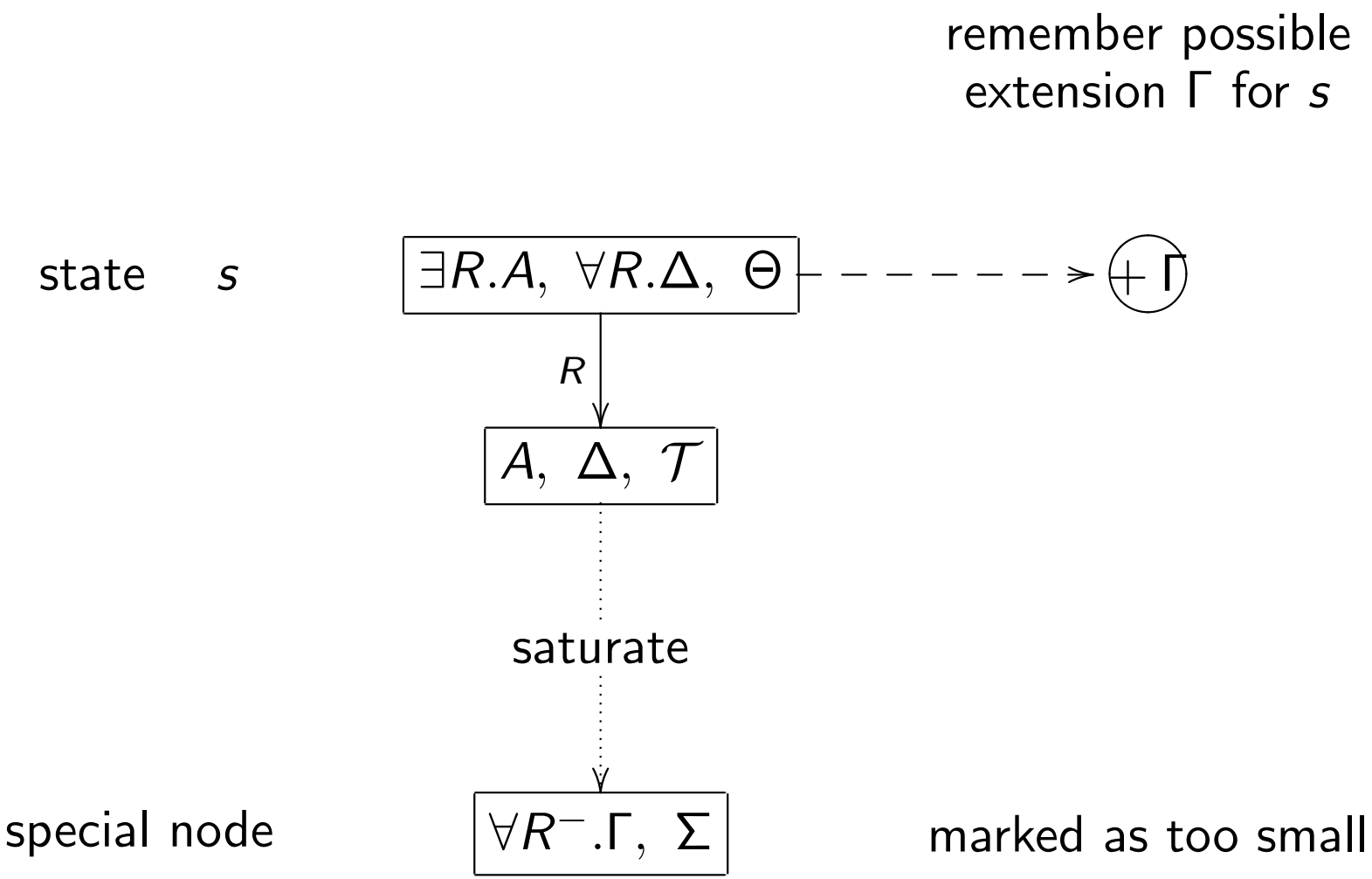


If Γ is contained in the formulae of s, the special node is *compatible* with s.

Compatible Special Nodes



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 an Or-Node

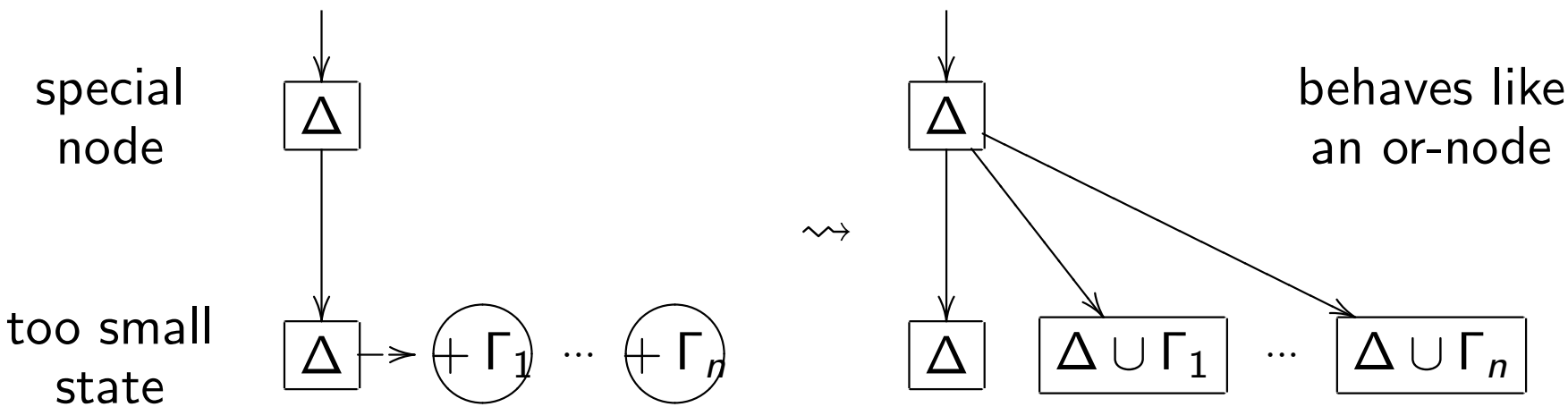
- some child is sat $\rightsquigarrow$ or-node is sat
- else: some child is open or unexpanded $\rightsquigarrow$ or-node is open
- else: some child is too small $\rightsquigarrow$ or-node is too small
- else: all children unsat $\rightsquigarrow$ or-node is unsat

Determining the Status of a State

- 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

- Optimality is a consequence of the fact that states are unique.
- If a node is sat or remains open, its formulae are jointly satisfiable.
- If a node is unsat, its formulae are not jointly satisfiable.
- If the root node is too small, its formulae are not jointly satisfiable.

Implications of Global State Caching

Termination: immediate since every pre-state must give a state

EXPTIME worst-case complexity: immediate

(Anywhere) Equality State Blocking: comes for free

(Global) `sat`/`unsat` caching: just keep a separate cache

Sound (Global) `sat` State caching: comes for free

Individual reuse: comes for free ... when it is saturated

Dynamic Blocking: not needed

Experimental Results

Implementation: in OCaml using some optimisations

Problems: randomly generated formulae of increasing sizes

Comparison: against FaCT++

Conclusion: they both behave similarly

Paper: "Sound global caching for ALC with Inverse Role", RG and Florian Widmann, Proc. TABLEAUX 2009

Bug in FaCT++: due to complexity of anywhere pair-wise blocking

Experimental Results

Implementation: in OCaml using some optimisations

Problems: randomly generated formulae of increasing sizes

Comparison: against FaCT++

Conclusion: they both behave similarly

Paper: "Sound global caching for ALC with Inverse Role", RG and Florian Widmann, Proc. TABLEAUX 2009

Bug in FaCT++: due to complexity of anywhere pair-wise blocking

Lecture 4: Fix-point Logics

LTL: linear temporal logic

CTL: computation tree logic

PDL: propositional dynamic logic

LCK: logic of common knowledge

CTL: Computation Tree Logic

Atomic Formulae: $p ::= p_0 \mid p_1 \mid p_2 \mid \cdots$ (AP)

Formulae: (Fml)

$$\begin{aligned} \varphi ::= p &\mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \to \varphi && \text{(CPL)} \\ &\mid EX\varphi \mid AX\varphi && \exists/\forall\text{-ne}\mathbf{X}\text{t} \\ &\mid E(\varphi\, U\, \psi) \mid A(\varphi\, U\, \psi) && \exists/\forall\text{-Until} \\ &\mid E(\varphi\, B\, \psi) \mid A(\varphi\, B\, \psi) && \exists/\forall\text{-Before} \end{aligned}$$

Note: Ep is not a formula!

Unary Modal connectives are: $EX\cdot$ and $AX\cdot$

Binary Modal Connectives are: $E(\cdot\, U\, \cdot)$ $A(\cdot\, U\, \cdot)$ $A(\cdot\, B\, \cdot)$ $E(\cdot\, B\, \cdot)$

NNF: later assume that all formulae are in Negation Normal Form

Semantics of CTL

Transition Frame: pair (W, R) where W is a non-empty set (of worlds) and R is a binary relation over W that is total/serial (which means that $\forall w \in W.\, \exists v \in W.\, w\, R\, v$)

Fullpath: in transition frame (W, R) is an infinite sequence $\sigma_0, \sigma_1, \sigma_2, \ldots$ of worlds in W with $\sigma_i\, R\, \sigma_{i+1}$ for all $i \in \mathbb{N}$

$\mathcal{B}(w)$: is the set of all fullpaths in (W, R) which begin at $w \in W$

Model: $M = (W, R, L)$ is a transition frame (W, R) and a labelling function $L : W \to 2^{\mathrm{AP}}$ where $L(w)$ is the set of atomic formulae true at w

Seriality: $\mathcal{B}(w)$ is non-empty by seriality

Semantics of CTL

Model: $M = (W, R, L)$ is a transition frame (W, R) and a labelling function $L : W \to 2^{\mathrm{AP}}$ where $L(w)$ is the set of atomic formulae true at w

World forces formula: defined by induction on shape of formula

$$
\begin{array}{lll}
M, w \Vdash p & \text{iff} & p \in L(w), \text{ for } p \in \mathrm{AP} \\
M, w \Vdash \neg\psi & \text{iff} & M, w \nVdash \psi \\
M, w \Vdash \varphi \wedge \psi & \text{iff} & M, w \Vdash \varphi \;\&\; M, w \Vdash \psi \\
M, w \Vdash \varphi \vee \psi & \text{iff} & M, w \Vdash \varphi \text{ or } M, w \Vdash \psi
\end{array}
$$

Intuition: classical connectives behave as usual at a world

Semantics of CTL

Model: $M = (W, R, L)$ is a transition frame (W, R) and a labelling function $L : W \to 2^{\mathrm{AP}}$ where $L(w)$ is the set of atomic formulae true at w

World forces formula: defined by induction on shape of formula

$$M, w \Vdash EX\varphi \quad \text{iff} \quad \exists v \in W.\, w\, R\, v \;\&\; M, v \Vdash \varphi$$

$$M, w \Vdash AX\varphi \quad \text{iff} \quad \forall v \in W.\, w\, R\, v \;\Rightarrow\; M, v \Vdash \varphi$$

Intuitions: $EX\varphi$ means "some immediate R-successor forces φ"

Intuitions: $AX\varphi$ means "every immediate R-successor forces φ"

X: stands for neXt i.e. immediate

Semantics of CTL

Model: $M = (W, R, L)$ is a transition frame (W, R) and a labelling function $L : W \to 2^{\mathrm{AP}}$ where $L(w)$ is the set of atomic formulae true at w

World forces formula: defined by induction on shape of formula

$M, w \Vdash E(\varphi \, U \, \psi)$ iff "some fullpath from w forces φ until ψ"

$M, w \Vdash A(\varphi \, U \, \psi)$ iff "every fullpath from w forces φ until ψ"

But: have not defined what it means for a fullpath to force a formula

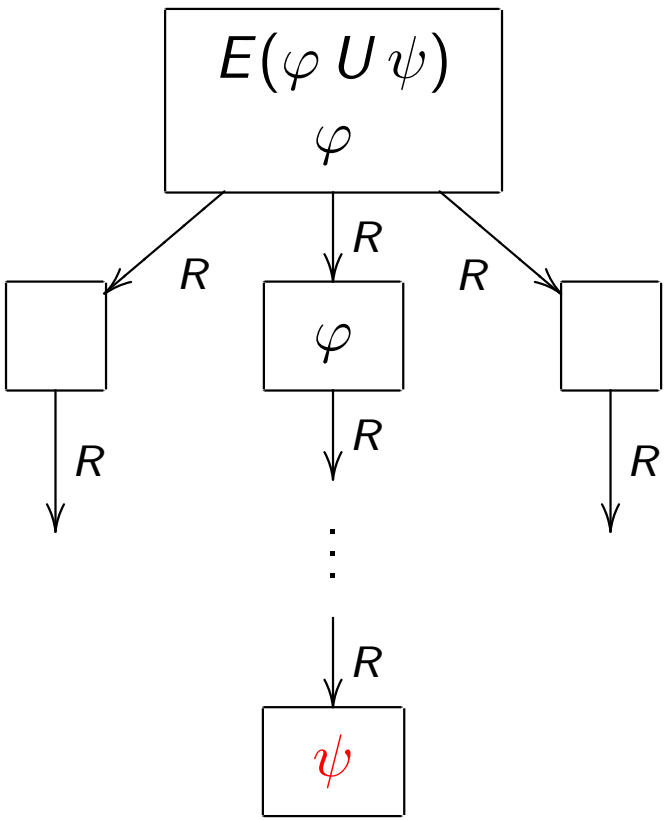
Must: express it in terms of a world forcing a formula

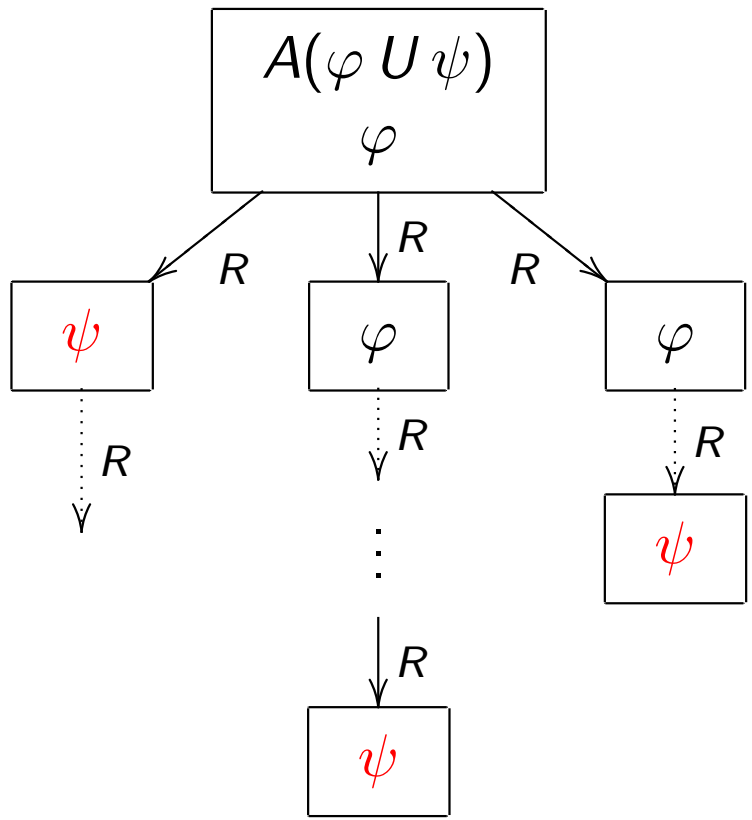
Semantics of CTL

World forces formula: defined by induction on shape of formula

$$M, w \Vdash E(\varphi\, U\, \psi) \quad \text{iff} \quad \exists \sigma \in \mathcal{B}(w).\ \exists i \in \mathbb{N}.\ [M, \sigma_i \Vdash \psi\ \&\ \forall j < i.\ M, \sigma_j \Vdash \varphi]$$

$$M, w \Vdash A(\varphi\, U\, \psi) \quad \text{iff} \quad \forall \sigma \in \mathcal{B}(w).\ \exists i \in \mathbb{N}.\ [M, \sigma_i \Vdash \psi\ \&\ \forall j < i.\ M, \sigma_j \Vdash \varphi]$$





Semantics of CTL

Model: $M = (W, R, L)$ is a transition frame (W, R) and a labelling function $L : W \to 2^{\mathrm{AP}}$ where $L(w)$ is the set of atomic formulae true at w

World forces formula: defined by induction on shape of φ

$$M, w \Vdash E(\varphi \, B \, \psi) \quad \text{iff} \quad \exists \sigma \in \mathcal{B}(w). \; \forall i \in \mathbb{N}. \, [M, \sigma_i \Vdash \psi \;\Rightarrow\; \exists j < i. \, M, \sigma_j \Vdash \varphi]$$

"some fullpath from w forces φ before it forces ψ"

$$M, w \Vdash A(\varphi \, B \, \psi) \quad \text{iff} \quad \forall \sigma \in \mathcal{B}(w). \; \forall i \in \mathbb{N}. \, [M, \sigma_i \Vdash \psi \;\Rightarrow\; \exists j < i. \, M, \sigma_j \Vdash \varphi]$$

"every fullpath from w forces φ before it forces ψ"

Note: it is possible that ψ is never forced in both cases

Exercises for CTL

1. Show that $M, w \Vdash AX\varphi$ iff $M, w \Vdash \neg EX\neg\varphi$ (duality)
2. Give semantics for $EF\varphi := E(\top\, U\, \varphi)$ where $\top := p_0 \vee \neg p_0$
3. Give semantics for $AF\varphi := A(\top\, U\, \varphi)$ where $\top := p_0 \vee \neg p_0$
4. Work out the semantics for $AG\varphi := \neg EF\neg\varphi$
5. Work out the semantics for $EG\varphi := \neg AF\neg\varphi$
6. Why can't we define $AG\varphi := A(\varphi\, U\, \bot)$ where $\bot := p_0 \wedge \neg p_0$
7. Why can't we define $EG\varphi := E(\varphi\, U\, \bot)$ where $\bot := p_0 \wedge \neg p_0$
8. Express $AG\varphi$ and $EG\varphi$ in terms of $A(\cdot\, B\, \cdot)$ and $E(\cdot\, B\, \cdot)$ (resp)

Exercises for CTL

NNF: Show that $\neg E(\varphi\, U\, \psi) \leftrightarrow A((\neg\varphi)\, B\, \psi)$ is CTL-valid

NNF: Show that $\neg A(\varphi\, U\, \psi) \leftrightarrow E((\neg\varphi)\, B\, \psi)$ is CTL-valid

NNF: Show that $\neg E(\varphi\, B\, \psi) \leftrightarrow A((\neg\varphi)\, U\, \psi)$ is CTL-valid

NNF: Show that $\neg A(\varphi\, B\, \psi) \leftrightarrow E((\neg\varphi)\, U\, \psi)$ is CTL-valid

β: Show that $E(p\, U\, q) \leftrightarrow q \vee (p \wedge EXE(p\, U\, q))$ is CTL-valid

β: Show that $A(p\, U\, q) \leftrightarrow q \vee (p \wedge AXA(p\, U\, q))$ is CTL-valid

α: Show that $E(p\, B\, q) \leftrightarrow \neg q \wedge (p \vee EXE(p\, B\, q))$ is CTL-valid

α: Show that $A(p\, B\, q) \leftrightarrow \neg q \wedge (p \vee AXA(p\, B\, q))$ is CTL-valid

Smullyan's $\alpha-$ and $\beta-$notation

Notation captures abstract "conjunctive" and "disjunctive" notions

α	α_1	α_2
$\varphi \wedge \psi$	φ	ψ
$E(\varphi\, B\, \psi)$	$\sim\psi$	$\varphi \vee EXE(\varphi\, B\, \psi)$
$A(\varphi\, B\, \psi)$	$\sim\psi$	$\varphi \vee AXA(\varphi\, B\, \psi)$
$AG\ \varphi$	φ	$AXAG\ \varphi$

β	β_1	β_2
$\varphi \vee \psi$	φ	ψ
$E(\varphi\, U\, \psi)$	ψ	$\varphi \wedge EXE(\varphi\, U\, \psi)$
$A(\varphi\, U\, \psi)$	ψ	$\varphi \wedge AXA(\varphi\, U\, \psi)$
$EF\ \varphi$	φ	$EXEF\ \varphi$

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

Prop: all instances of $\alpha \leftrightarrow \alpha_1 \wedge \alpha_2$ and $\beta \leftrightarrow \beta_1 \vee \beta_2$ are CTL-valid

Note: some of these equivalences require that R is serial/total

Tableau Rules for CTL

$\boldsymbol{\alpha}$	$\boldsymbol{\alpha}_1$	$\boldsymbol{\alpha}_2$
$\varphi \wedge \psi$	φ	ψ
$E(\varphi\, B\, \psi)$	$\sim\psi$	$\varphi \vee EXE(\varphi\, B\, \psi)$
$A(\varphi\, B\, \psi)$	$\sim\psi$	$\varphi \vee AXA(\varphi\, B\, \psi)$
$AG\ \varphi$	φ	$AXAG\ \varphi$

$\boldsymbol{\beta}$	$\boldsymbol{\beta}_1$	$\boldsymbol{\beta}_2$
$\varphi \vee \psi$	φ	ψ
$E(\varphi\, U\, \psi)$	ψ	$\varphi \wedge EXE(\varphi\, U\, \psi)$
$A(\varphi\, U\, \psi)$	ψ	$\varphi \wedge AXA(\varphi\, U\, \psi)$
$EF\ \varphi$	φ	$EXEF\ \varphi$

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

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

Assume: that all formulae are in Negation Normal Form

Tableau Rules: only applicable to non-starred formulae

$$(\boldsymbol{\alpha})\frac{\Gamma; \boldsymbol{\alpha}}{\boldsymbol{\alpha}^*; \Gamma; \boldsymbol{\alpha}_1; \boldsymbol{\alpha}_2} \qquad (\boldsymbol{\beta})\frac{\Gamma; \boldsymbol{\beta}}{\boldsymbol{\beta}^*; \Gamma; \boldsymbol{\beta}_1 \mid \boldsymbol{\beta}^*; \Gamma; \boldsymbol{\beta}_2} \qquad (EX)\frac{\Lambda; EX\varphi_1; \cdots; EX\varphi_n; AX\Delta}{\varphi_1; \Delta \,||\, \cdots \,||\, \varphi_n; \Delta}$$

where Λ contains only atoms, negated atoms and starred formulae

Tableau Calculus for CTL: (Construction) Phase 1

$\langle V, E \rangle$: a graph of nodes V and edges E

Initialise: put $V := \{r\}$ where root node r contains a given finite set Y of NNF-formulae, put $E := \emptyset$ and mark r as unexpanded

Repeat: choose an unexpanded node x from V

Expand: If some rule is applicable to x then

Apply: apply the rule giving $n \geq 1$ children $w_1, \cdots, w_n$

Check: for $i = 1 \cdots n$, if w_i duplicates some $v \in V$ then add (x, v) to E else add w_i to V and add (x, w_i) to E

Mark: x as expanded

Until all nodes in V are expanded

Seriality: for every childless node $x \in V$ add (x, x) to E

Termination: only finite number of different nodes possible

Tableau Method for CTL: (Pruning) Phase 2

Unstar: starred formulae in $\langle V, E \rangle$

Path: a maximal (cyclic) sequence of nodes starting at the root

Eventuality: $E(\varphi \, U \, \psi)$ / $A(\varphi \, U \, \psi)$ since it entails that eventually ψ must become true on some/every path

Fulfilled: $E(\varphi \, U \, \psi) \in s$ is fulfilled if for some path $s_0 = s, s_1, \cdots$ from s there exists a k such that $\psi \in s_k$ and $\varphi \in s_j$ for all $j < k$

Fulfilled: $A(\varphi \, U \, \psi) \in s$ is fulfilled if for every path $s_0 = s, s_1, \cdots$ from s there exists a k such that $\psi \in s_k$ and $\varphi \in s_j$ for all $j < k$

Repeat: given $\langle V, E \rangle$ from Phase 1

- delete all nodes that contain a pair $\{p, \neg p\}$
- delete all nodes with no E-successor (seriality)
- delete all nodes that contain an un-fulfilled eventuality

Until: root is deleted or no node is deleted

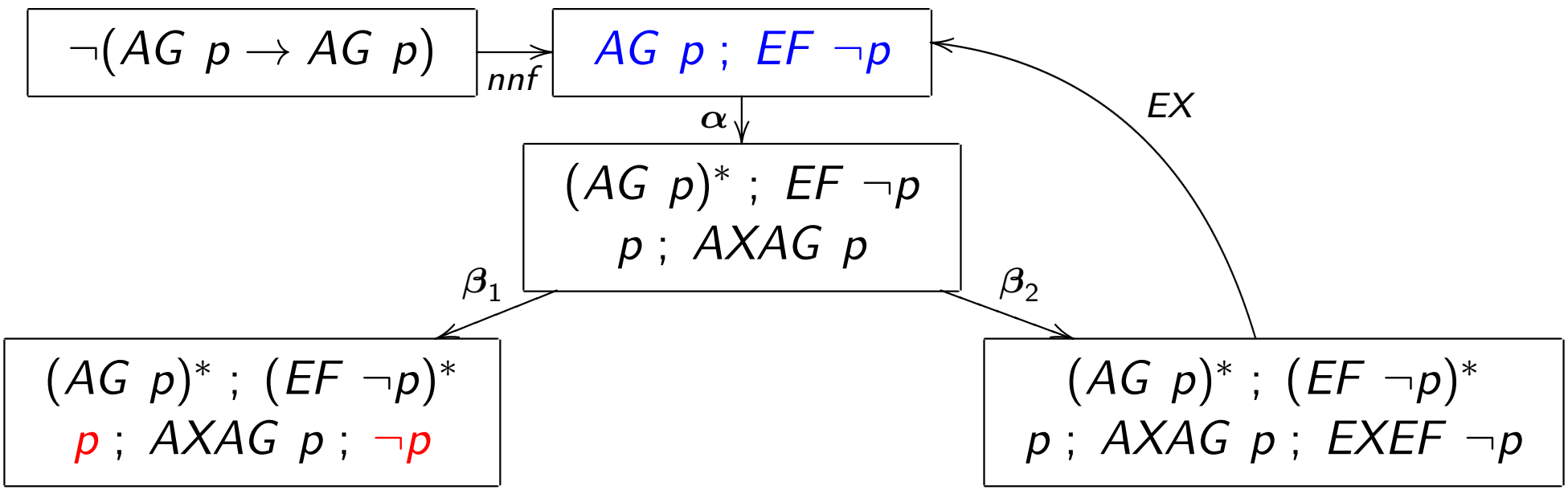
Thm: Y is satisfiable iff the root node is not deleted

Example: is $AGp \rightarrow AGp$ CTL-valid ? Phase 1

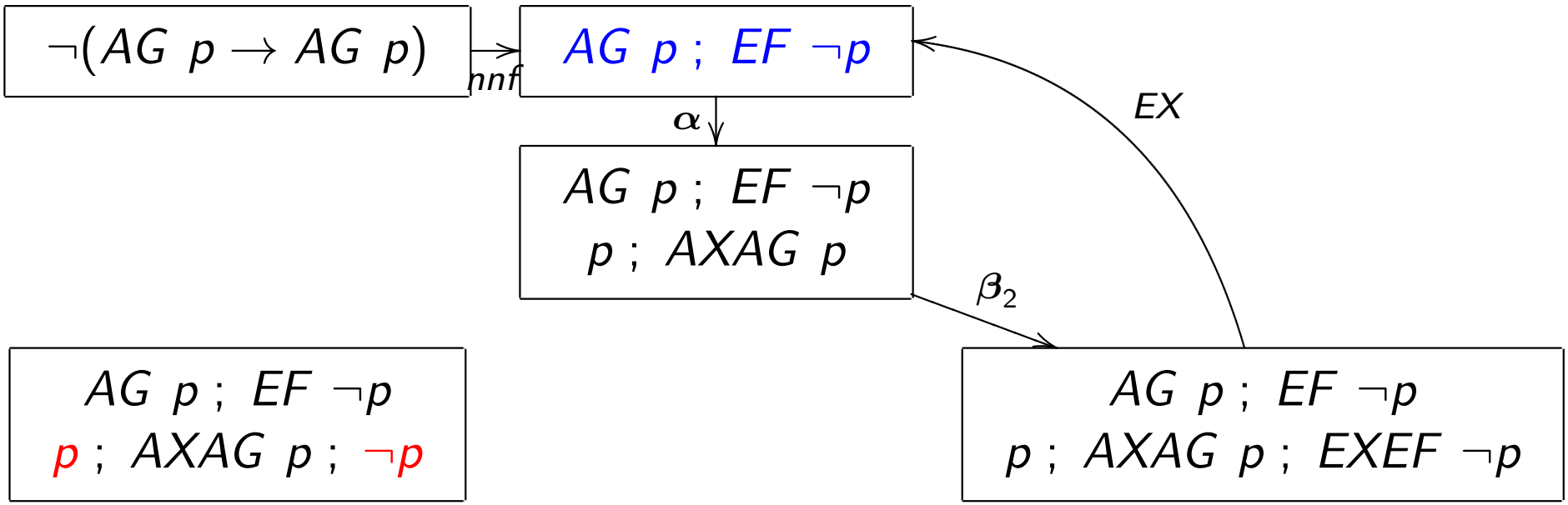
α	α_1	α_2
$AG\ \varphi$	φ	$AXAG\ \varphi$

β	β_1	β_2
$EF\ \varphi$	φ	$EXEF\ \varphi$

$$(EX)\frac{\Lambda; EX\varphi_1; \cdots ; EX\varphi_n; AX\Delta}{\varphi_1; \Delta \mid\mid \cdots \mid\mid \varphi_n; \Delta}$$



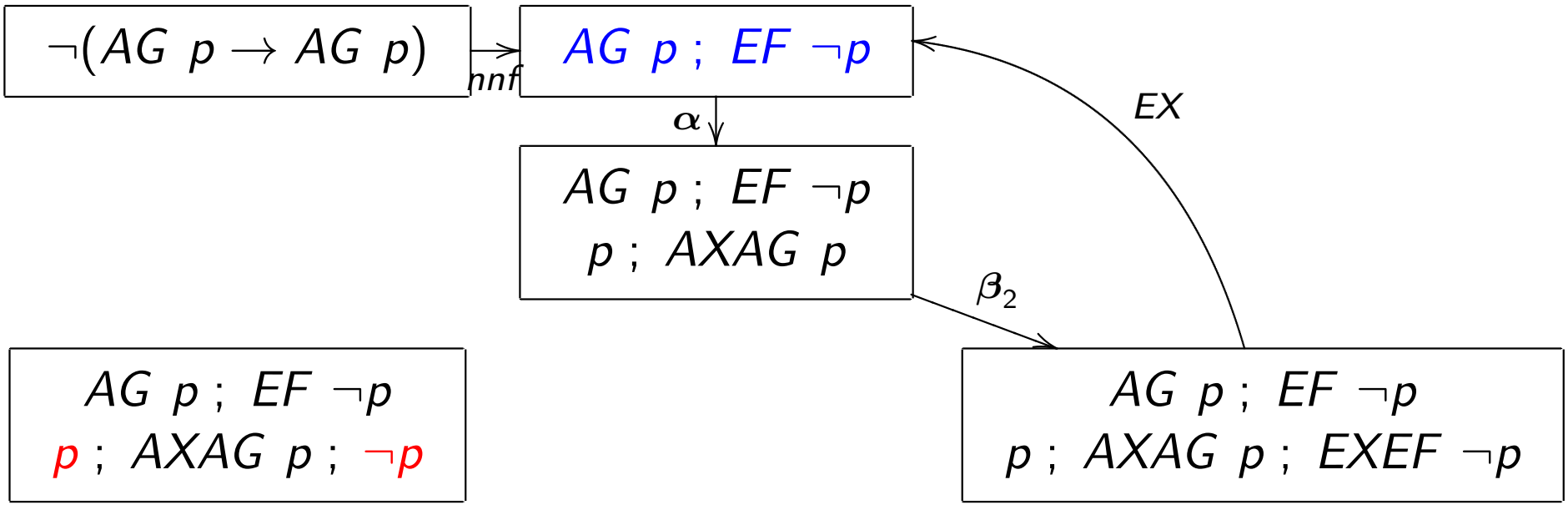
Example: is $AGp \rightarrow AGp$ CTL-valid ? Phase 2



Unstar all starred formulae

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

Example: is $AGp \rightarrow AGp$ CTL-valid ? Phase 2



Prune the root containing $EF\ \neg p$ since no path reaches $\neg p$

That is, $AG\ p\ ;\ EF\ \neg p$ is not CTL-satisfiable.

Hence $AG\ p \rightarrow AG\ p$ is CTL-valid.

Fixpoint Logics: PLTL, CTL, LCK, PDL

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

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

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

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

Least Fixpoint Expansion Rule: now or later

Global condition: Must ensure that the formula ψ eventually becomes true (unfulfilled eventualities))

Traditional Multipass Methods

Phase 1: apply rules to obtain a cyclic graph

Phase 2: prune inconsistent nodes and nodes that cannot fulfil their eventualities using multiple passes

Answer `unsat`: if root node gets pruned

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

Advantage: optimal

Disadvantage: can do unnecessary work

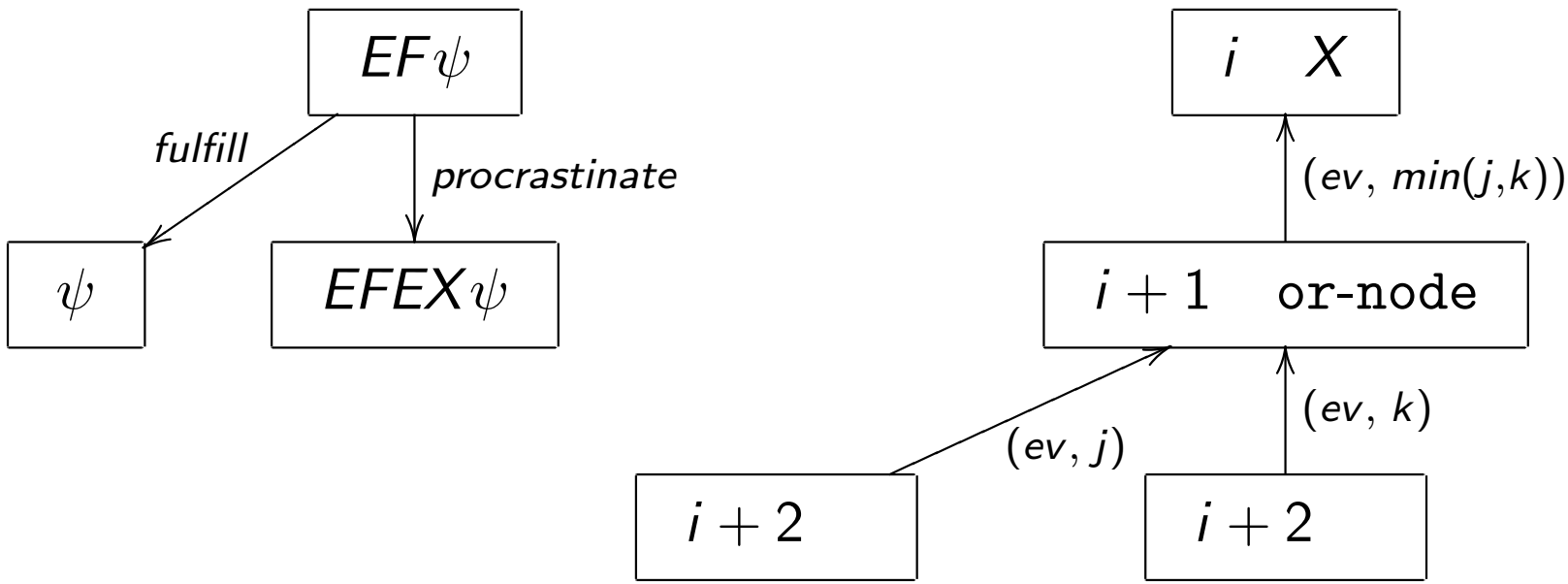
Example: $EF\psi_1 \wedge EF\Psi_2$ where $EF\psi$ is easily shown to be unsatisfiable and $EF\Psi$ is much harder

One-pass And-Or Tree Tableaux

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)



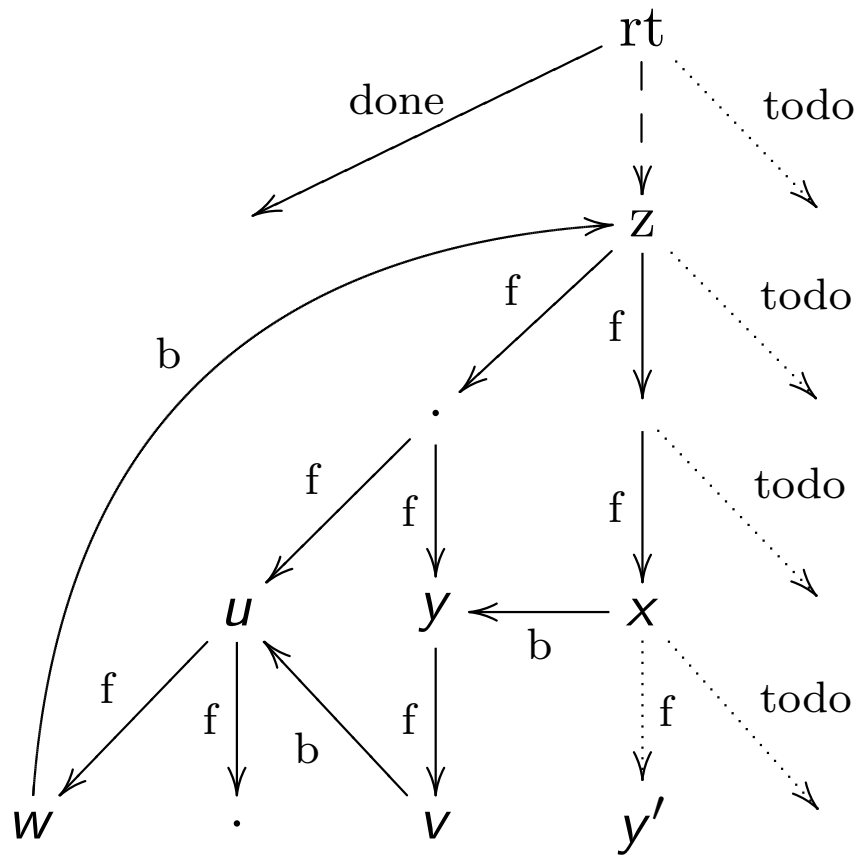
Close: any node at height h with an eventuality $(ev, h+1)$

On-the-fly And-Or Graph Tableaux

Interleave: the graph building and graph pruning phases

Advantage: optimal

Disadvantage: requires more memory



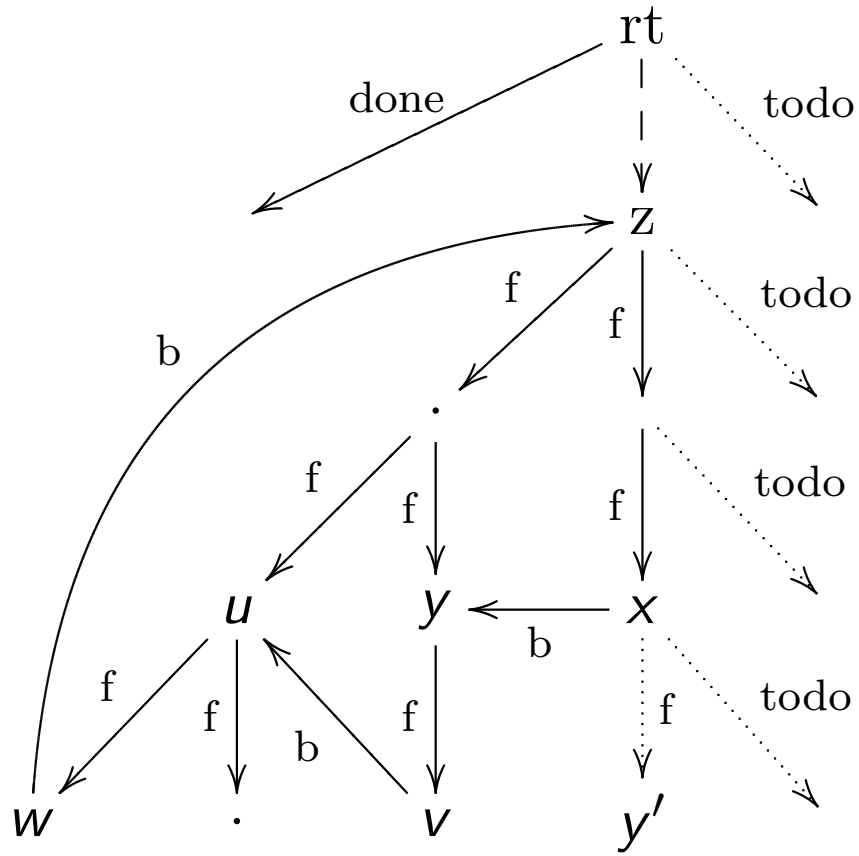
On-the-fly And-Or Graph Tableaux

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

Closed: unfulfilled and unfulfillable



Theorems for On-the-fly Tableau

Thm: root.status $=$ `unsat` implies X is unsatisfiable wrt $\mathcal{T}$

Proof: not trivial

Thm: root.status $=$ `sat` implies X is satisfiable wrt $\mathcal{T}$

Proof: not trivial

Thm: Worst-case complexity is EXPTIME

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

Experimental Results

GMUL: multipass method using and-or tree-tableaux

BDDCTL: BBD-based method

CTL-RP: modal resolution-based method

TreeTab: tree-tableaux uev-based method (suboptimal)

MLSolver: method based upon parity games

Conclusions:

- resolution and bdd methods are more robust
- TreeTab either succeeds spectacularly or fails spectacularly
- GMUL is not competitive against CTL-RP and BDDCTL
- MLSolver is generally worst
- hybrid of TreeTab and BBDCTL solves the most problems

Further work: needed to find efficient data structures for tableau-based methods (as already enjoyed by BDD and resolution methods)

Experimental Results

GMUL: multipass method using and-or tree-tableaux

BDDCTL: BBD-based method

CTL-RP: modal resolution-based method

TreeTab: tree-tableaux uev-based method (suboptimal)

MLSolver: method based upon parity games

Conclusions:

- resolution and bdd methods are more robust
- TreeTab either succeeds spectacularly or fails spectacularly
- GMUL is not competitive against CTL-RP and BDDCTL
- MLSolver is generally worst
- hybrid of TreeTab and BBDCTL solves the most problems

Further work: needed to find efficient data structures for tableau-based methods (as already enjoyed by BDD and resolution methods)

Tableaux-based Prototypes and a Warning

Provers for many logics available:

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

Warning: Obvious optimisations not always sound

References

ALC: "Sound Global Caching for ALC", DL 2007

ALC "An Experimental Evaluation of Global Caching for ALC (System Description)", IJCAR 2008

ALCI: "Sound Global State Caching for ALC with Inverse Roles", TABLEAUX 2009

PLTL: "A New One-Pass Tableau Algorithm for PLTL", S Schwendimann, Proc. TABLEAUX 1998

CTL: "One-pass Tableaux for Computational Tree Logic", Proc. LPAR 2007

LCK: "Cut-free Single-pass Tableaux for the Logic of Common Knowledge" Workshop on Agents and Deduction at TABLEAUX 2007

PDL: "An On-the-fly Tableau-based Decision Procedure for PDL", Proc. M4M 2007

PDL: "An Optimal On-the-fly Tableau-based Decision Procedure for PDL", CADE 2009

CPDL: "An Optimal and Cut-free Tableau-based Decision Procedure for CPDL", IJCAR 2010