

Ch. 13 Tableaux systems

Andreas Herzig

IRIT-CNRS

Toulouse

<http://www.irit.fr/~Andreas.Herzig/>

Overview

- tableaux systems: basic ideas
- tableaux systems: basic definitions
- tableaux for simple modal logics
- tableaux for transitive modal logics
- tableaux for modal logics with transitive closure and other modal logics
- some implemented tableaux theorem provers

Conventions

- monomodal logic, modal operator \Box
 - formulas A, B, \dots
- $M = (W, R, V)$, where $V: (W \times \text{Atm}) \rightarrow \{0, 1\}$
 - $V_w(p) = 1, V_w(q) = 0$

Overview

- **tableaux systems: basic ideas**
- tableaux systems: basic definitions
- tableaux for simple modal logics
- tableaux for transitive modal logics
- tableaux for modal logics with transitive closure and other modal logics
- some implemented tableaux theorem provers

The basic idea for classical logic [Beth]

- given formula A , try to find M and w by applying the truth conditions (“tableau rules”)

$w \Vdash A \wedge B \quad \rightarrow$ add $w \Vdash A$, and add $w \Vdash B$

$w \Vdash A \vee B \quad \rightarrow$ add either $w \Vdash A$, or add $w \Vdash B$ (nondet.)

$w \Vdash \sim A \quad \rightarrow$ “don’t add $w \Vdash A$ ” ???

– $w \Vdash \sim \sim A \quad \rightarrow$ add $w \Vdash A$

– $w \Vdash \sim(A \vee B) \quad \rightarrow$ add $w \Vdash \sim A$, and add $w \Vdash \sim B$

– $w \Vdash \sim(A \wedge B) \quad \rightarrow$ add either $w \Vdash \sim A$, or add $w \Vdash \sim B$

- apply while possible (“downwards saturation”)
- is this a model?

NO if both $w \Vdash P$ and $w \Vdash \sim P$ (“tableau is closed”)

ELSE: for every w , if $w \Vdash P$ put $V_w(P) = 1$, else put $V_w(P) = 0$

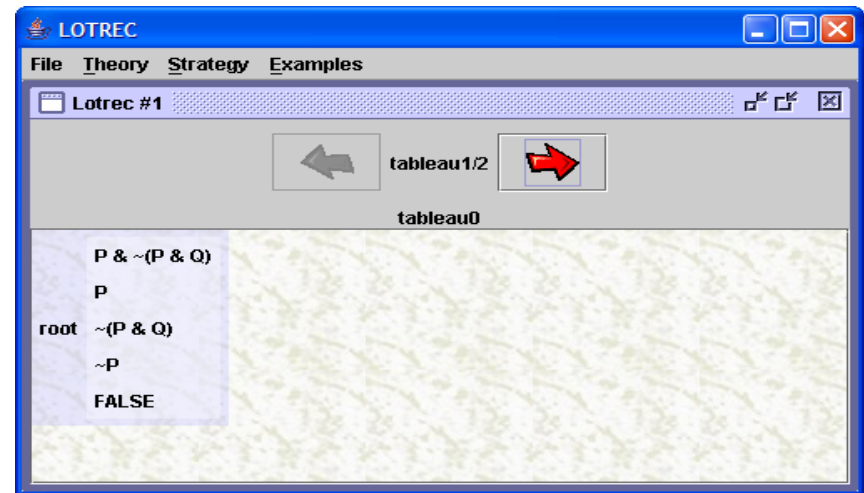
The basic idea: example for classical logic

$$A = P \wedge \sim(P \wedge Q)$$

- applying truth conditions:
 1. $w \Vdash P \wedge \sim(P \wedge Q)$
 2. $w \Vdash P \wedge \sim(P \wedge Q)$, $w \Vdash P$, $w \Vdash \sim(P \wedge Q)$
 3. $w \Vdash P \wedge \sim(P \wedge Q)$, $w \Vdash P$, $w \Vdash \sim(P \wedge Q)$, $w \Vdash \sim P$ (nondet.)
- no more truth condition applies
- can't be a model:
both $w \Vdash P$ and $w \Vdash \sim P$
- backtrack on nondeterministic choices

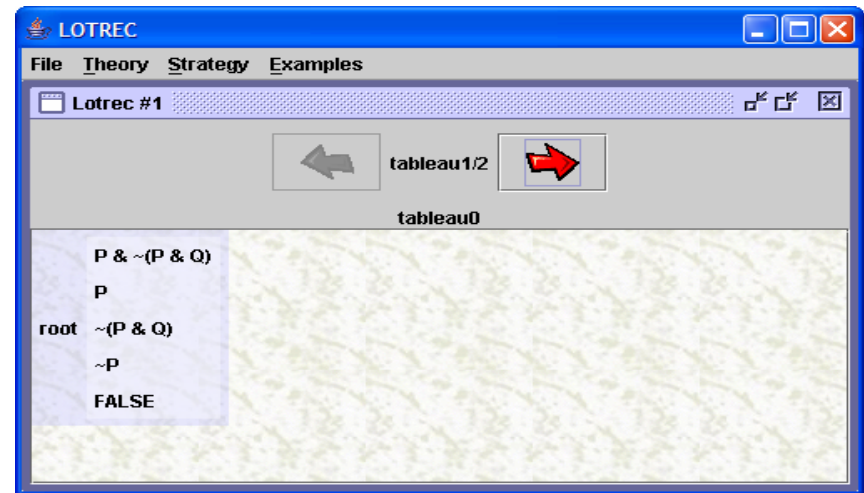
The basic idea: example for classical logic (ctd.)

- 1st downward saturated graph for
 $A = P \wedge \sim(P \wedge Q)$
→ not a model
(contains P and $\sim P!$)

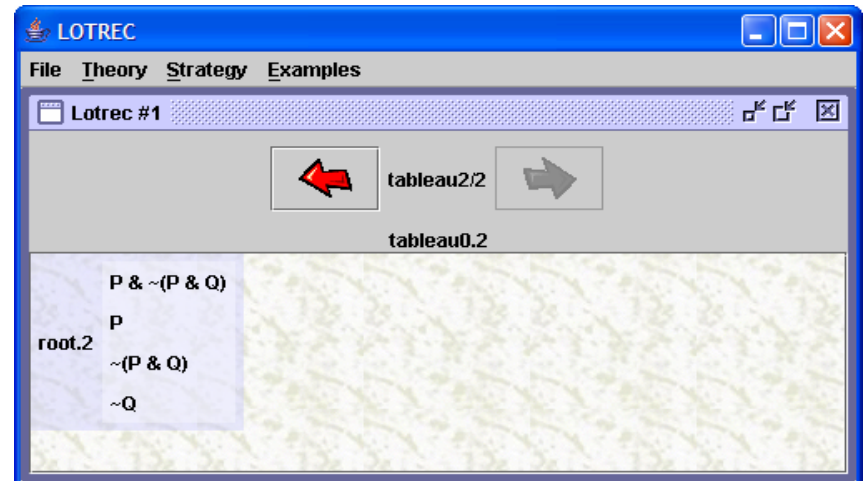


The basic idea: example for classical logic (ctd.)

- 1st downward saturated set for
 $A = P \wedge \sim(P \wedge Q)$
→ not a model
(contains P and $\sim P$!)



- 2nd downward saturated set for
 $A = P \wedge \sim(P \wedge Q)$
→ is a model of A



The basic idea for modal logics

- apply truth conditions = build a graph
 - create nodes
 - add links between nodes
 - add formulas to nodes

- the basic cases

$w \Vdash \Box A$ \rightarrow for all u such that Rwu , add $u \Vdash A$

$w \Vdash \Diamond A$ \rightarrow add some new u , add Rwu , add $u \Vdash A$

$w \Vdash \sim \Box A$ \rightarrow add some new u , add Rwu , add $u \Vdash \sim A$

$w \Vdash \sim \Diamond A$ \rightarrow ...

- “downwards saturated graph”: is this a model?

The basic idea: example for modal logic

$$A = P \wedge \sim \Box P$$

- applying tableau rules:
 1. $w \Vdash P \wedge \sim \Box P$
 2. $w \Vdash P \wedge \sim \Box P, w \Vdash P, w \Vdash \sim \Box P$
 3. $w \Vdash P \wedge \sim \Box P, w \Vdash P, w \Vdash \sim \Box P, R w u, u \Vdash \sim P$no more tableau rule applies
→ never both $w \Vdash A$ and $w \Vdash \sim A$ (“open tableau”)
- model can be built: $M = (W, R, V)$

set of worlds W : $W = \{w, u\}$
accessibility relation R : $R \ni w u$
valuation V : $V_w(P) = 1, V_u(P) = 0$

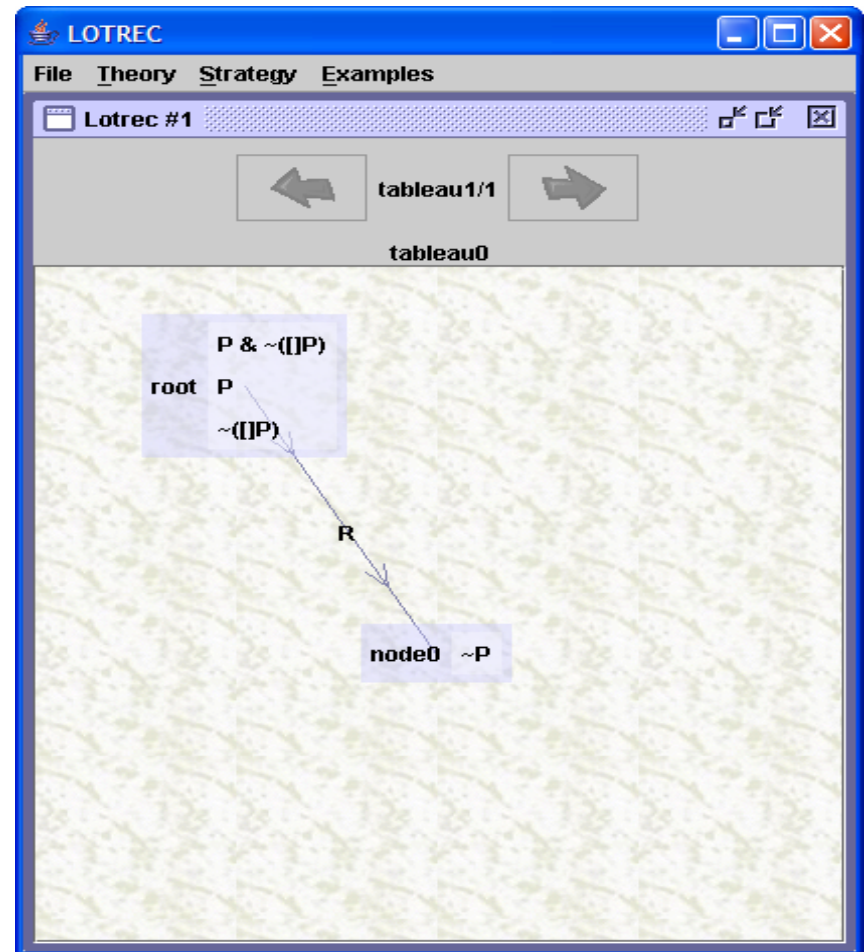
The basic idea: example for modal logic (ctd.)

- premodel for

$$A = P \wedge \sim \Box P$$

→ not closed

→ is a model of A



A remark on tableaux and truth tables

- Tableaux are a more convenient presentation of the familiar truth table analysis” [Beth]
- “Tableaux are more efficient than truth tables.” [folklore]
- ... not exactly [d’Agostino]:
 - $(P1 \vee P2 \vee P3) \wedge (P1 \vee P2 \vee \sim P3) \wedge (P1 \vee \sim P2 \vee P3) \wedge \dots$
 - there are formulas with n atoms of length $O(2^n)$
 - truth tables have 2^n rows
 - at least $n!$ closed tableaux, and $n!$ grows faster than 2^n

Historical remarks

- the early days (1950-80): handwritten proofs
 - Beth, Gentzen
 - relation to sequent calculus
 - “tableau proof = sequent proof backwards”
 - Kripke: explicit accessibility relation
 - Smullyan, Fitting: uniform notation
- today: mechanized systems
 - fast provers exist
 - FaCT [Horrocks]
 - K-SAT [Giunchiglia&Sebastiani]
 - importance of strategies
 - applications exist: BDI logics, description logics

Overview

- possible worlds semantics: quickstart
- tableaux systems: basic ideas
- **tableaux systems: basic definitions**
- tableaux for simple modal logics
- tableaux for transitive modal logics
- tableaux for intuitionistic logic
- tableaux for other nonclassical logics
- tableaux for modal logics with transitive closure and other modal and description logics
- tableaux for 1st order logic
- some implemented tableaux theorem provers

Informal definition of tableau rules

- Tableau rules expand directed graphs by
 - adding formulas
 - adding nodes
 - adding links
 - duplicating the graph
- $\text{rule}(G) = \{G_1, \dots, G_n\}$

Informal definition of tableau rules

- Tableau rules expand directed graphs by
 - adding formulas
 - adding nodes
 - adding links
 - duplicating the graph
- $\text{rule}(G) = \{G_1, \dots, G_n\}$
- application of a rule to $G =$
application to *every* formula in *every* node of G .
- $\text{rule}(\{G_1, \dots, G_n\}) = \text{rule}(G_1) \cup \dots \cup \text{rule}(G_n)$

Tableau rules: syntax

- general form:

```
rule ruleName
  if cond1
  ...
  if condn
  do action1
  ...
  do actionk
```

- example conditions:

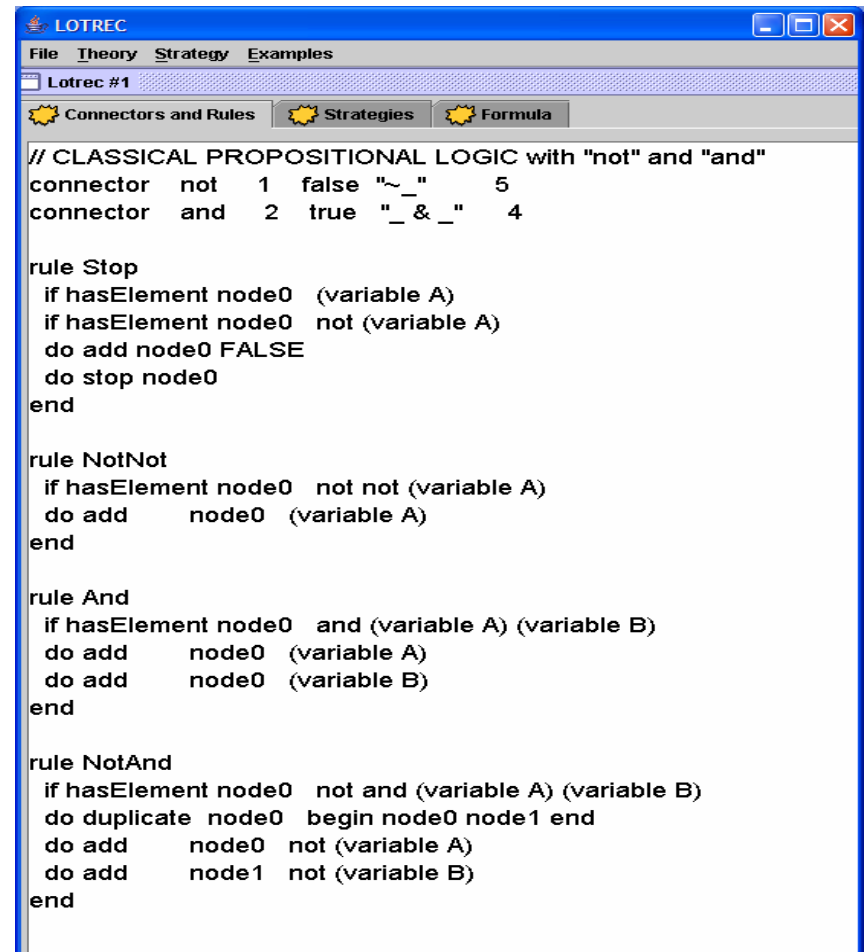
```
if hasElement node formula
if isLinked node1 node2 R
... (more to come)
```

- example actions:

```
do stop
do addElement node formula
do newNode node
do link node1 node2 R
do duplicate node1 [...]
... (more to come)
```

Example: tableau rules for classical logic

the
LoTREC
tableau
prover



```
LOTREC
File Theory Strategy Examples
Lotrec #1
Connectors and Rules Strategies Formula
// CLASSICAL PROPOSITIONAL LOGIC with "not" and "and"
connector not 1 false "~_" 5
connector and 2 true "_&_" 4

rule Stop
  if hasElement node0 (variable A)
  if hasElement node0 not (variable A)
  do add node0 FALSE
  do stop node0
end

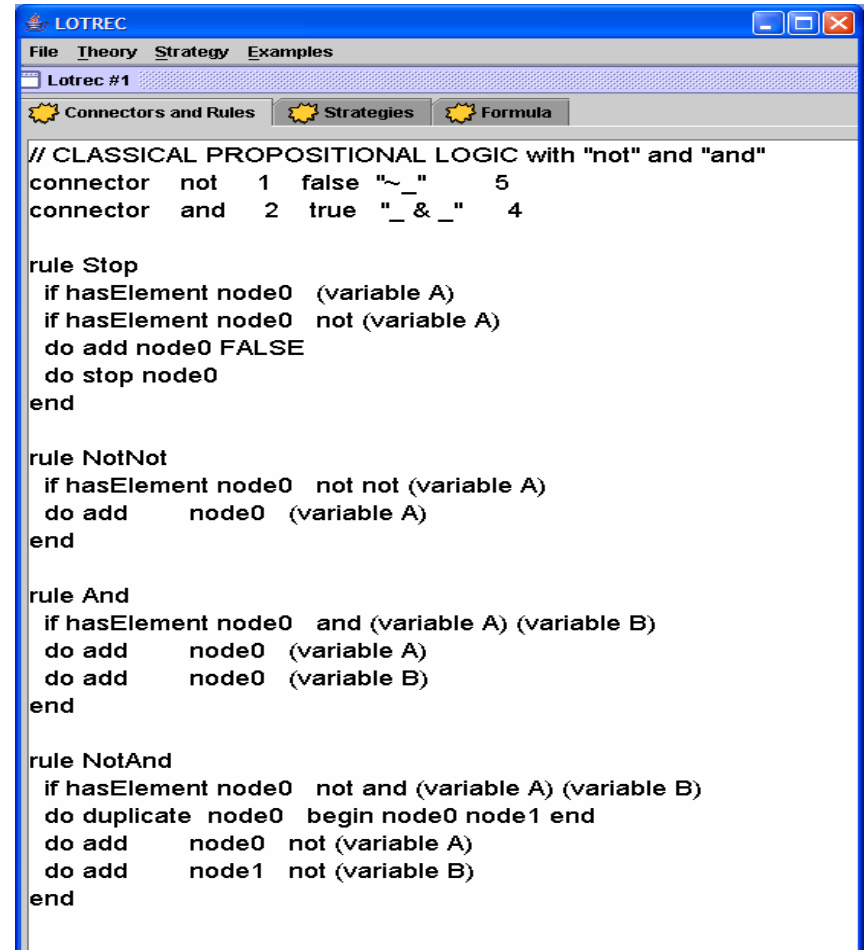
rule NotNot
  if hasElement node0 not not (variable A)
  do add node0 (variable A)
end

rule And
  if hasElement node0 and (variable A) (variable B)
  do add node0 (variable A)
  do add node0 (variable B)
end

rule NotAnd
  if hasElement node0 not and (variable A) (variable B)
  do duplicate node0 begin node0 node1 end
  do add node0 not (variable A)
  do add node1 not (variable B)
end
```

Example: tableau rules for classical logic

declaration of connectors:
negation and conjunction only

The image shows a screenshot of the LOTREC software interface. The window title is "LOTREC" and it has a menu bar with "File", "Theory", "Strategy", and "Examples". Below the menu bar, there are three tabs: "Connectors and Rules", "Strategies", and "Formula". The "Connectors and Rules" tab is active, and it displays the following text:

```
// CLASSICAL PROPOSITIONAL LOGIC with "not" and "and"
connector not 1 false "~_" 5
connector and 2 true "_&_" 4

rule Stop
  if hasElement node0 (variable A)
  if hasElement node0 not (variable A)
  do add node0 FALSE
  do stop node0
end

rule NotNot
  if hasElement node0 not not (variable A)
  do add node0 (variable A)
end

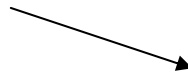
rule And
  if hasElement node0 and (variable A) (variable B)
  do add node0 (variable A)
  do add node0 (variable B)
end

rule NotAnd
  if hasElement node0 not and (variable A) (variable B)
  do duplicate node0 begin node0 node1 end
  do add node0 not (variable A)
  do add node1 not (variable B)
end
```

An arrow points from the text "declaration of connectors: negation and conjunction only" to the first two lines of the code in the screenshot.

Example: tableau rules for classical logic

rule Stop:
if there is an explicit contradiction
then stop exploring the tableau



```
LOTREC
File Theory Strategy Examples
Lotrec #1
Connectors and Rules Strategies Formula
// CLASSICAL PROPOSITIONAL LOGIC with "not" and "and"
connector not 1 false "~_" 5
connector and 2 true "_ &_" 4

rule Stop
  if hasElement node0 (variable A)
  if hasElement node0 not (variable A)
  do add node0 FALSE
  do stop node0
end

rule NotNot
  if hasElement node0 not not (variable A)
  do add node0 (variable A)
end

rule And
  if hasElement node0 and (variable A) (variable B)
  do add node0 (variable A)
  do add node0 (variable B)
end

rule NotAnd
  if hasElement node0 not and (variable A) (variable B)
  do duplicate node0 begin node0 node1 end
  do add node0 not (variable A)
  do add node1 not (variable B)
end
```

Example: tableau rules for classical logic

rule NotNot:
replaces $\sim\sim A$ by A



```
LOTREC
File Theory Strategy Examples
Lotrec #1
Connectors and Rules Strategies Formula
// CLASSICAL PROPOSITIONAL LOGIC with "not" and "and"
connector not 1 false "~_" 5
connector and 2 true "_&_" 4

rule Stop
  if hasElement node0 (variable A)
  if hasElement node0 not (variable A)
  do add node0 FALSE
  do stop node0
end

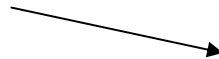
rule NotNot
  if hasElement node0 not not (variable A)
  do add node0 (variable A)
end

rule And
  if hasElement node0 and (variable A) (variable B)
  do add node0 (variable A)
  do add node0 (variable B)
end

rule NotAnd
  if hasElement node0 not and (variable A) (variable B)
  do duplicate node0 begin node0 node1 end
  do add node0 not (variable A)
  do add node1 not (variable B)
end
```

Example: tableau rules for classical logic

rule And:
if A & B is in a node
then add A and B to node



```
LOTREC
File Theory Strategy Examples
Lotrec #1
Connectors and Rules Strategies Formula
// CLASSICAL PROPOSITIONAL LOGIC with "not" and "and"
connector not 1 false "~_" 5
connector and 2 true "_&_" 4

rule Stop
  if hasElement node0 (variable A)
  if hasElement node0 not (variable A)
  do add node0 FALSE
  do stop node0
end

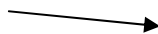
rule NotNot
  if hasElement node0 not not (variable A)
  do add node0 (variable A)
end

rule And
  if hasElement node0 and (variable A) (variable B)
  do add node0 (variable A)
  do add node0 (variable B)
end

rule NotAnd
  if hasElement node0 not and (variable A) (variable B)
  do duplicate node0 begin node0 node1 end
  do add node0 not (variable A)
  do add node1 not (variable B)
end
```

Example: tableau rules for classical logic

rule NotAnd:
if $\sim(A \& B)$ is in a node
then duplicate tableau,
add $\sim A$ to the first tableau
add $\sim B$ to the second tableau



```
LOTREC
File Theory Strategy Examples
Lotrec #1
Connectors and Rules Strategies Formula
// CLASSICAL PROPOSITIONAL LOGIC with "not" and "and"
connector not 1 false "~_" 5
connector and 2 true "_&_" 4

rule Stop
  if hasElement node0 (variable A)
  if hasElement node0 not (variable A)
  do add node0 FALSE
  do stop node0
end

rule NotNot
  if hasElement node0 not not (variable A)
  do add node0 (variable A)
end

rule And
  if hasElement node0 and (variable A) (variable B)
  do add node0 (variable A)
  do add node0 (variable B)
end

rule NotAnd
  if hasElement node0 not and (variable A) (variable B)
  do duplicate node0 begin node0 node1 end
  do add node0 not (variable A)
  do add node1 not (variable B)
end
```

Definition of strategies

- A *strategy* defines some order of application of the tableau rules:
 - firstrule $rule_1 \dots rule_n$ end
“apply first applicable rule and stop”
 - allrules $rule_1 \dots rule_n$ end
“apply all applicable rules in order”
 - repeat *strategy* end
“repeat until no rule applicable”
- Strategy stops if no rule is applicable.

Strategy for classical logic

strategy CPLStrategy

repeat allRules

Stop

NotNot

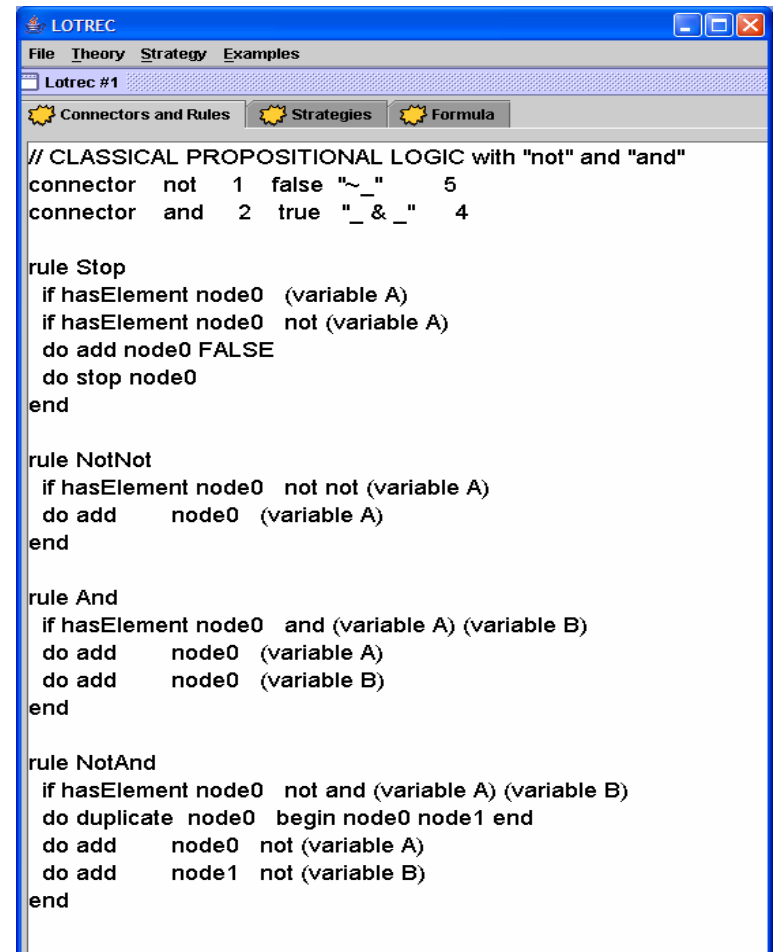
And

NotAnd

end end

end

→ “fair strategy”



```
LOTREC
File Theory Strategy Examples
Lotrec #1
Connectors and Rules Strategies Formula
// CLASSICAL PROPOSITIONAL LOGIC with "not" and "and"
connector not 1 false "~_" 5
connector and 2 true "_&_" 4

rule Stop
  if hasElement node0 (variable A)
  if hasElement node0 not (variable A)
  do add node0 FALSE
  do stop node0
end

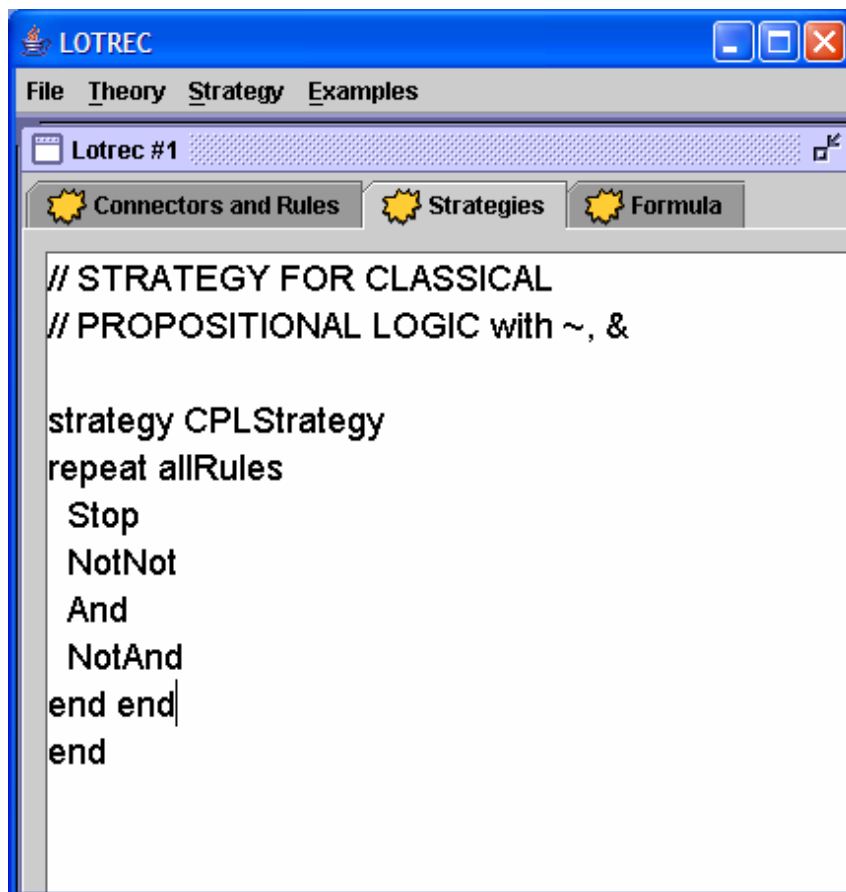
rule NotNot
  if hasElement node0 not not (variable A)
  do add node0 (variable A)
end

rule And
  if hasElement node0 and (variable A) (variable B)
  do add node0 (variable A)
  do add node0 (variable B)
end

rule NotAnd
  if hasElement node0 not and (variable A) (variable B)
  do duplicate node0 begin node0 node1 end
  do add node0 not (variable A)
  do add node1 not (variable B)
end
```

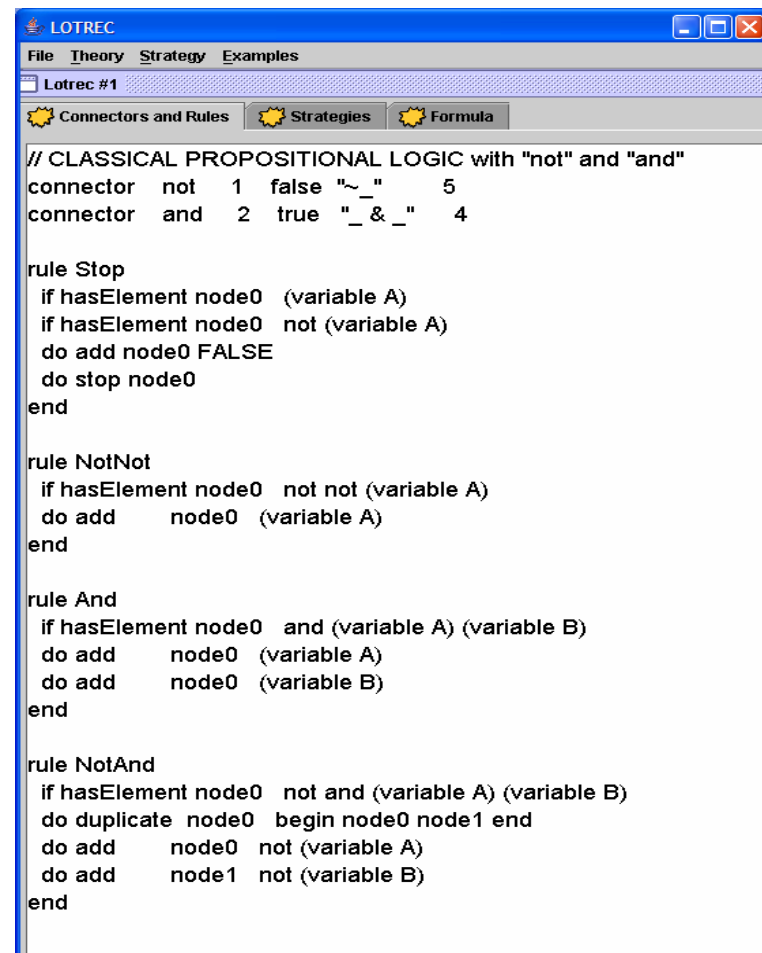
Strategy for classical logic: example

CPLStrategy(P&~(P&Q))



```
LOTREC
File Theory Strategy Examples
Lotrec #1
Connectors and Rules Strategies Formula
// STRATEGY FOR CLASSICAL
// PROPOSITIONAL LOGIC with ~, &

strategy CPLStrategy
repeat allRules
  Stop
  NotNot
  And
  NotAnd
end end
end
```



```
LOTREC
File Theory Strategy Examples
Lotrec #1
Connectors and Rules Strategies Formula
// CLASSICAL PROPOSITIONAL LOGIC with "not" and "and"
connector not 1 false "~_" 5
connector and 2 true "_&_" 4

rule Stop
if hasElement node0 (variable A)
if hasElement node0 not (variable A)
do add node0 FALSE
do stop node0
end

rule NotNot
if hasElement node0 not not (variable A)
do add node0 (variable A)
end

rule And
if hasElement node0 and (variable A) (variable B)
do add node0 (variable A)
do add node0 (variable B)
end

rule NotAnd
if hasElement node0 not and (variable A) (variable B)
do duplicate node0 begin node0 node1 end
do add node0 not (variable A)
do add node1 not (variable B)
end
```

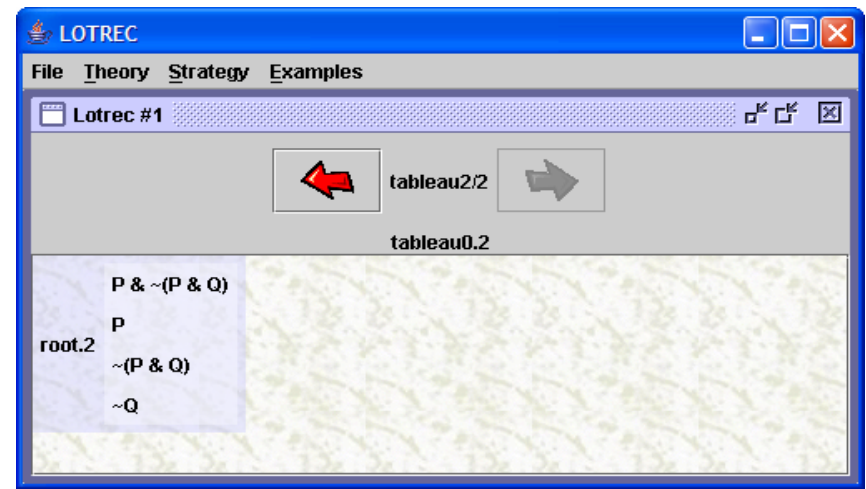
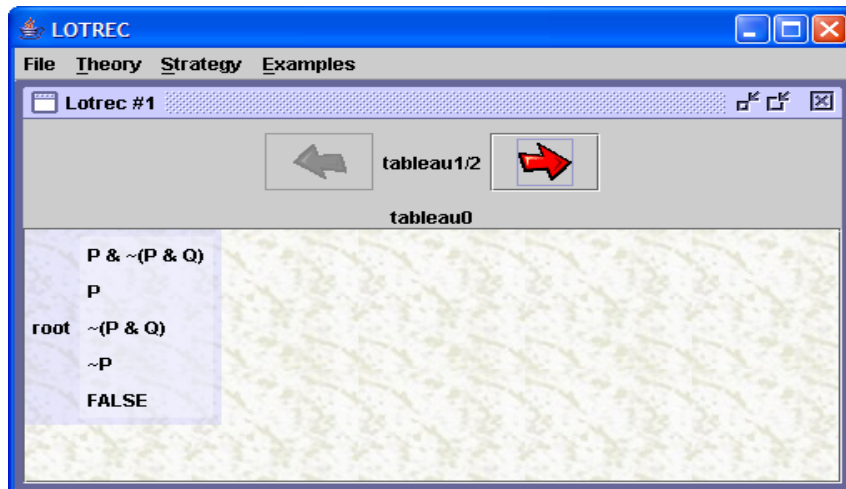
Strategy for classical logic: example (ctd.)

CPLStrategy($P \& \sim(P \& Q)$) =

{ T1

,

T2 }



Definition of tableaux

The *set of tableaux for A with strategy S* is the set of graphs obtained by applying the strategy S to an initial single-node graph whose root contains only A.

- notation: $S(A)$
 - Remark
our tableau = “tableau branch” in the literature
(sounds odd to call a graph a branch)

Tableaux: open or closed?

- *A node is closed* iff it contains FALSE.
- *A tableau is closed* iff it has a closed node.
- *A set of tableaux is closed*
iff all its elements are.

An open tableau is a premodel:

→ build a model

Formal properties

to be proved for each strategy:

- Termination

For every A , $S(A)$ terminates.

- Soundness

If $S(A)$ is *closed* then A is *unsatisfiable*.

- Completeness

If $S(A)$ is *open* then A is *satisfiable*.

Termination

- For every A , $\text{CPLTableaux}(A)$ terminates.
- Proof:
 - Every tableau rule only adds strict subformulas.
 - This can only be done a finite number of times, then the strategy stops.

Soundness

- If $\text{CPLTableaux}(A)$ is closed then A is unsatisfiable.
- Proof:
 - Every tableau rule is “guaranteed” by the truth conditions:
 - If G is CPL-satisfiable
 - then there is G_i in $\text{rule}(G)$ that is CPL-satisfiable
 - Hence if every graph is closed then the original A cannot be satisfiable.

Completeness

- If $\text{CPLTableaux}(A)$ is open then A is satisfiable.
- Proof:
 - Take some open tableau G in $\text{CPLTableaux}(A)$.

Completeness

- If $\text{CPLTableaux}(A)$ is open then A is satisfiable.
 - Proof:
 - Take some open tableau G in $\text{CPLTableaux}(A)$.
 - G is a downwards closed set (“Hintikka set”):
 - if $\sim\sim A$ in node then A in node
 - if $A\&B$ in node then A in node and B in node
 - if $\sim(A\&B)$ in node then $\sim A$ in node or $\sim B$ in node
- (because allRules strategy is fair: every rule eventually applies)

Completeness

- If $\text{CPLTableaux}(A)$ is open then A is satisfiable.
- Proof:
 - Take some open tableau G in $\text{CPLTableaux}(A)$.
 - G is a downwards closed set (“Hintikka set”):
 - if $\sim\sim A$ in node then A in node
 - if $A\&B$ in node then A in node and B in node
 - if $\sim(A\&B)$ in node then $\sim A$ in node or $\sim B$ in node(because allRules strategy is fair: every rule eventually applies)
 - Build a CPL model from G :
 - $V_{\text{node}}(P) = 1$ iff P appears in node

Completeness

- If $\text{CPLTableaux}(A)$ is open then A is satisfiable.
- Proof:
 - Take some open tableau G in $\text{CPLTableaux}(A)$.
 - G is a downwards closed set (“Hintikka set”):
 - if $\sim\sim A$ in node then A in node
 - if $A\&B$ in node then A in node and B in node
 - if $\sim(A\&B)$ in node then $\sim A$ in node or $\sim B$ in node(because allRules strategy is fair: every rule eventually applies)
 - Build a CPL model from G :
 - $V_{\text{node}}(P) = 1$ iff P appears in node
 - Prove by induction on the form of A :
 - for every A in node, $V_{\text{node}}(A) = 1$(“fundamental lemma”)

In general ...

- soundness proof ... easy
- termination proof ... difficult
- completeness proof ... very difficult

In general ...

- soundness proof: easy
- termination proof: difficult
- completeness proof: very difficult

- ... but soundness + termination of strategy is practically sufficient:
 1. apply strategy to A
 2. take an open tableau and build pointed model (M,w)
 3. check if M in model class
 4. check if $M,w \models A$

Overview

- possible worlds semantics: quickstart
- tableaux systems: basic ideas
- tableaux systems: basic definitions
- **tableaux for simple modal logics**
- tableaux for transitive modal logics
- tableaux for intuitionistic logic
- tableaux for other nonclassical logics
- tableaux for modal logics with transitive closure and other modal and description logics
- tableaux for 1st order logic
- some implemented tableaux theorem provers

The basic modal logic K

- the basic modal operators:

$w \Vdash \Box A$ iff for all u : Rwu implies $u \Vdash A$

$w \Vdash \Diamond A$ iff exists u : Rwu and $u \Vdash A$

Tableau rules for K

connectors: not, and, nec

[some rules for classical logic...]

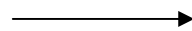
Tableau rules for K

connectors: not, and, nec

[some rules for classical logic...]

createSuccessor:

if not nec A is in node0
then create new node node1
link it to node0
add not A to node1



end

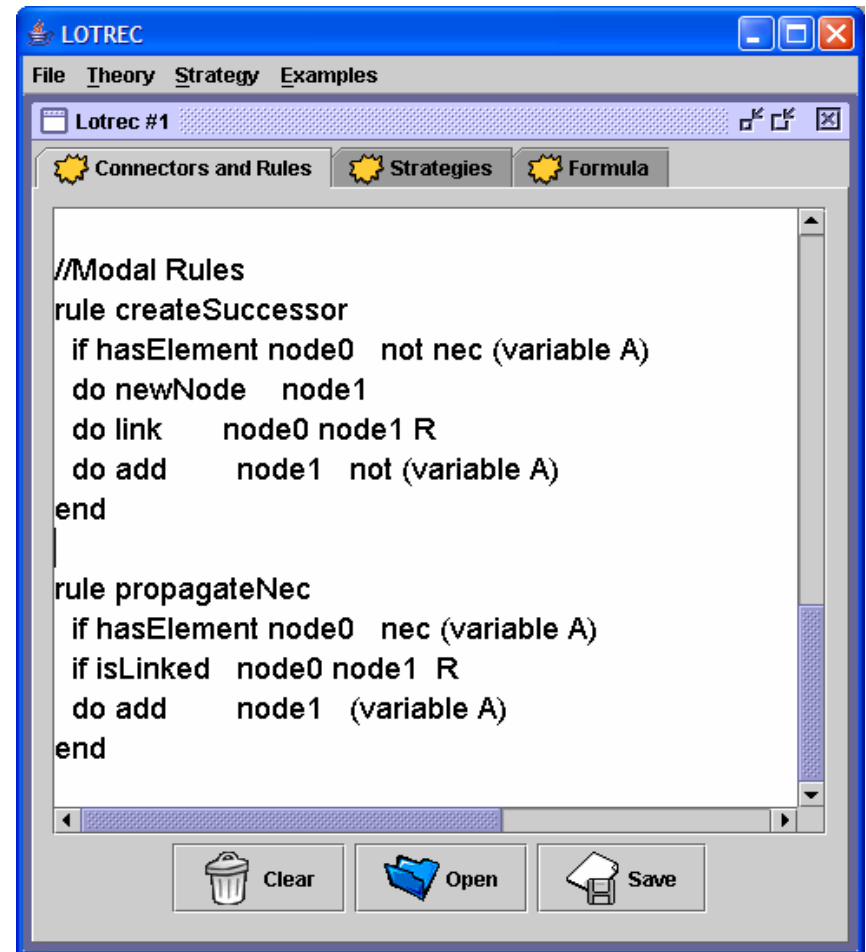


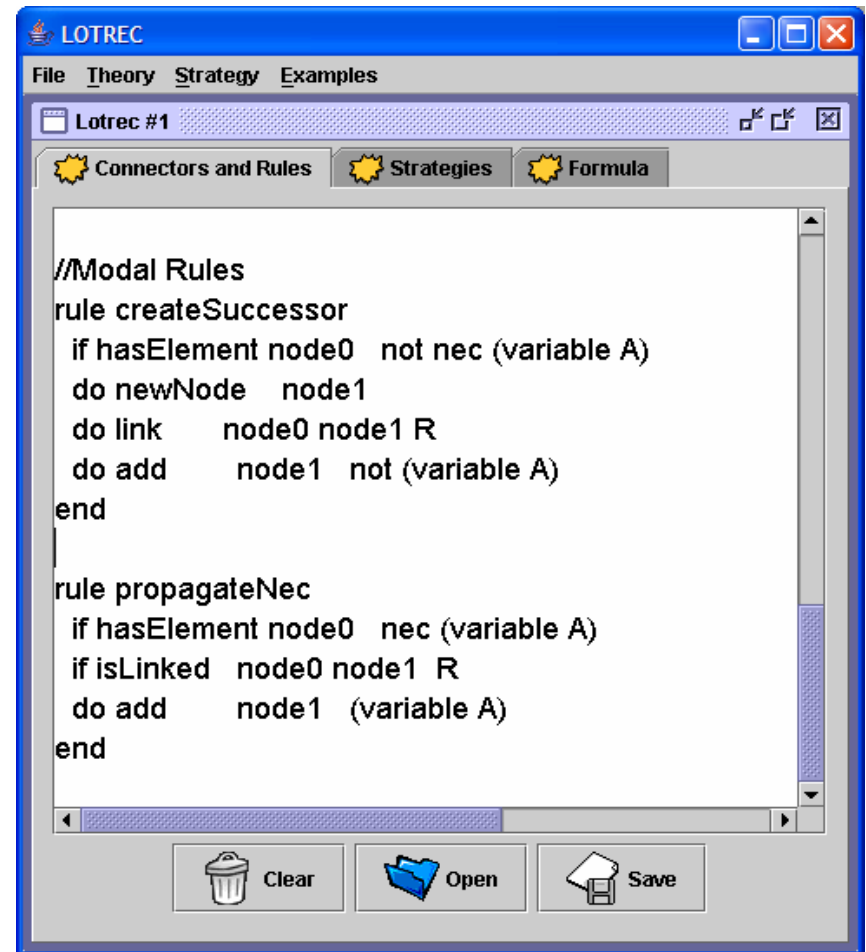
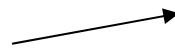
Tableau rules for K

connectors: not, and, nec

[some rules for classical logic...]

propagateNec:

```
if   nec A is in node0
    node0 is linkednode1 R
then add   node1 A
end
```



Tableaux for K

- ... plus rules for the definable connectives
- KStrategy($\leftrightarrow P$ & $\leftrightarrow Q$ & $\Box(R \vee \leftrightarrow S)$)

Modal logic KT

- accessibility relation is *reflexive*
- idea: integrate this into truth condition
 - $w \Vdash \Box A$ iff $w \Vdash A$ and for all u : Rwu implies $u \Vdash A$

Tableaux for modal logic KT

[connectors as for K...]

[rules as for K...]

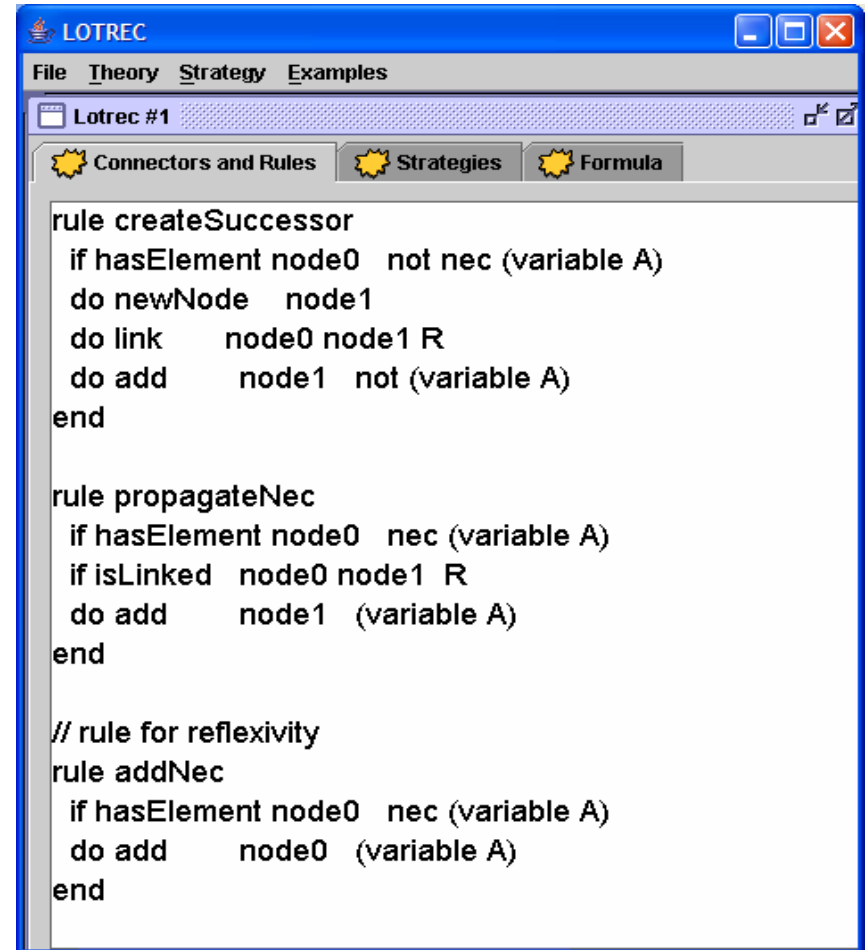
Tableaux for modal logic KT

[connectors as for K...]

[rules as for K...]

plus: “when $\Box A$ is in a node
then add A to it”

- KTStrategy(P & $\Box\Box\sim P$)



The screenshot shows the LOTREC software interface. The window title is "LOTREC" and the menu bar includes "File", "Theory", "Strategy", and "Examples". The main window displays a strategy definition for "Lotrec #1". The strategy is defined in three parts: "createSuccessor", "propagateNec", and "addNec".

```
rule createSuccessor
  if hasElement node0  not nec (variable A)
  do newNode  node1
  do link     node0 node1 R
  do add     node1  not (variable A)
end

rule propagateNec
  if hasElement node0  nec (variable A)
  if isLinked  node0 node1 R
  do add     node1  (variable A)
end

// rule for reflexivity
rule addNec
  if hasElement node0  nec (variable A)
  do add     node0  (variable A)
end
```

Tableaux for modal logic S5

accessibility relation is
equivalence relation

can be supposed to be
a single equivalence
class

optimized tableau rules

...

Overview

- tableaux systems: basic ideas
- tableaux systems: basic definitions
- tableaux for simple modal logics
- **tableaux for transitive modal logics**
- tableaux for modal logics with transitive closure and other modal logics
- some implemented tableaux theorem provers

Tableau rules for S4

accessibility relation is *reflexive* and *transitive*

tableau rules for S4:

- *[connectors as for KT...]*
- *[rules as for KT...]*
- ... and take into account transitivity:
“when $\Box A$ is in a node
then add $\Box A$ to all children”

Tableau rules for S4

accessibility relation is *reflexive* and *transitive*

tableau rules for S4:

- *[connectors as for KT...]*
- *[rules as for KT...]*
- ... and take into account transitivity:
“if $\Box A$ is in a node
then add $\Box A$ to all children”

problem: find a terminating strategy

Tableau rules for S4

- Example: $w \Vdash \Box \sim \Box P$

– add $w \Vdash \sim \Box P$

(by rule for reflexivity)

Tableau rules for S4

- Example: $w \Vdash \Box \sim \Box P$
 - add $w \Vdash \sim \Box P$ (by rule for reflexivity)
 - create u , add Rwu , add $u \Vdash \sim P$ (by createSuccessor)

Tableau rules for S4

- Example: $w \Vdash \Box \sim \Box P$
 - add $w \Vdash \sim \Box P$ (by rule for reflexivity)
 - create u , add Rwu , add $u \Vdash \sim P$ (by createSuccessor)
 - add $u \Vdash \Box \sim \Box P$ (by rule for transitivity)

Tableau rules for S4

- Example: $w \Vdash \Box \sim \Box P$
 - add $w \Vdash \sim \Box P$ (by rule for reflexivity)
 - create u , add Rwu , add $u \Vdash \sim P$ (by createSuccessor)
 - add $u \Vdash \Box \sim \Box P$ (by rule for transitivity)
 - add $u \Vdash \sim \Box P$ (by rule for reflexivity)

Tableau rules for S4

- Example: $w \Vdash \Box \sim \Box P$
 - add $w \Vdash \sim \Box P$ (by rule for reflexivity)
 - create u , add Rwu , add $u \Vdash \sim P$ (by createSuccessor)
 - add $u \Vdash \Box \sim \Box P$ (by rule for transitivity)
 - add $u \Vdash \sim \Box P$ (by rule for reflexivity)
 - create u'
 - ...

Tableau rules for S4

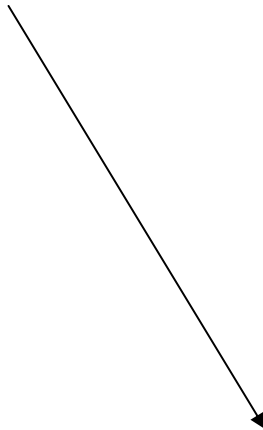
- Example: $w \Vdash \Box \sim \Box P$
 - add $w \Vdash \sim \Box P$ (by rule for reflexivity)
 - create u , add Rwu , add $u \Vdash \sim P$ (by createSuccessor)
 - add $u \Vdash \Box \sim \Box P$ (by rule for transitivity)
 - add $u \Vdash \sim \Box P$ (by rule for reflexivity)
 - create u'
 - ...

put a looptest into the rules!

Tableau rules for S4 (ctd.)

principle:

- if a node is *included* in an ancestor then mark it.



```
LOTREC
File Theory Strategy Examples
Lotrec #1
Connectors and Rules Strategies Formula

// rule for transitivity
rule copyNec
  if hasElement node0 nec (variable A)
  if isLinked node0 node1 R
  do add node1 nec (variable A)
end

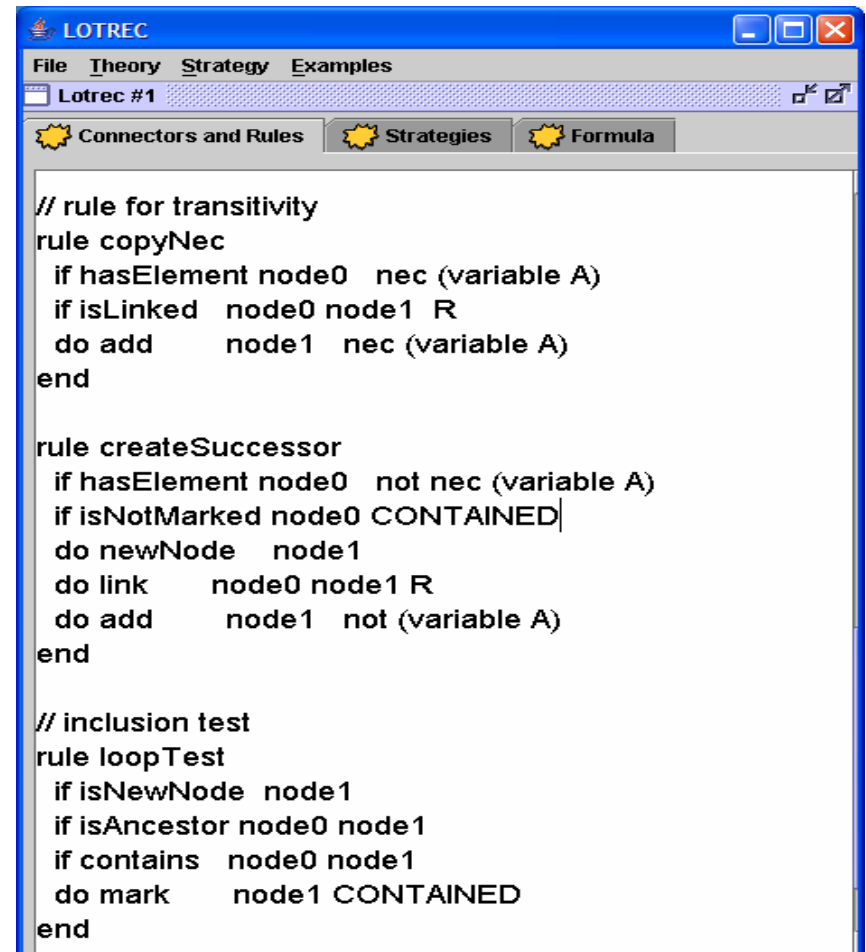
rule createSuccessor
  if hasElement node0 not nec (variable A)
  if isNotMarked node0 CONTAINED
  do newNode node1
  do link node0 node1 R
  do add node1 not (variable A)
end

// inclusion test
rule loopTest
  if isNewNode node1
  if isAncestor node0 node1
  if contains node0 node1
  do mark node1 CONTAINED
end
```

Tableau rules for S4 (ctd.)

principle:

- if a node is *included* in an ancestor then mark it.
- if a node is marked then block the createSuccessor rule
- [S4Strategy\(\[~\]\[P\]\)](#)



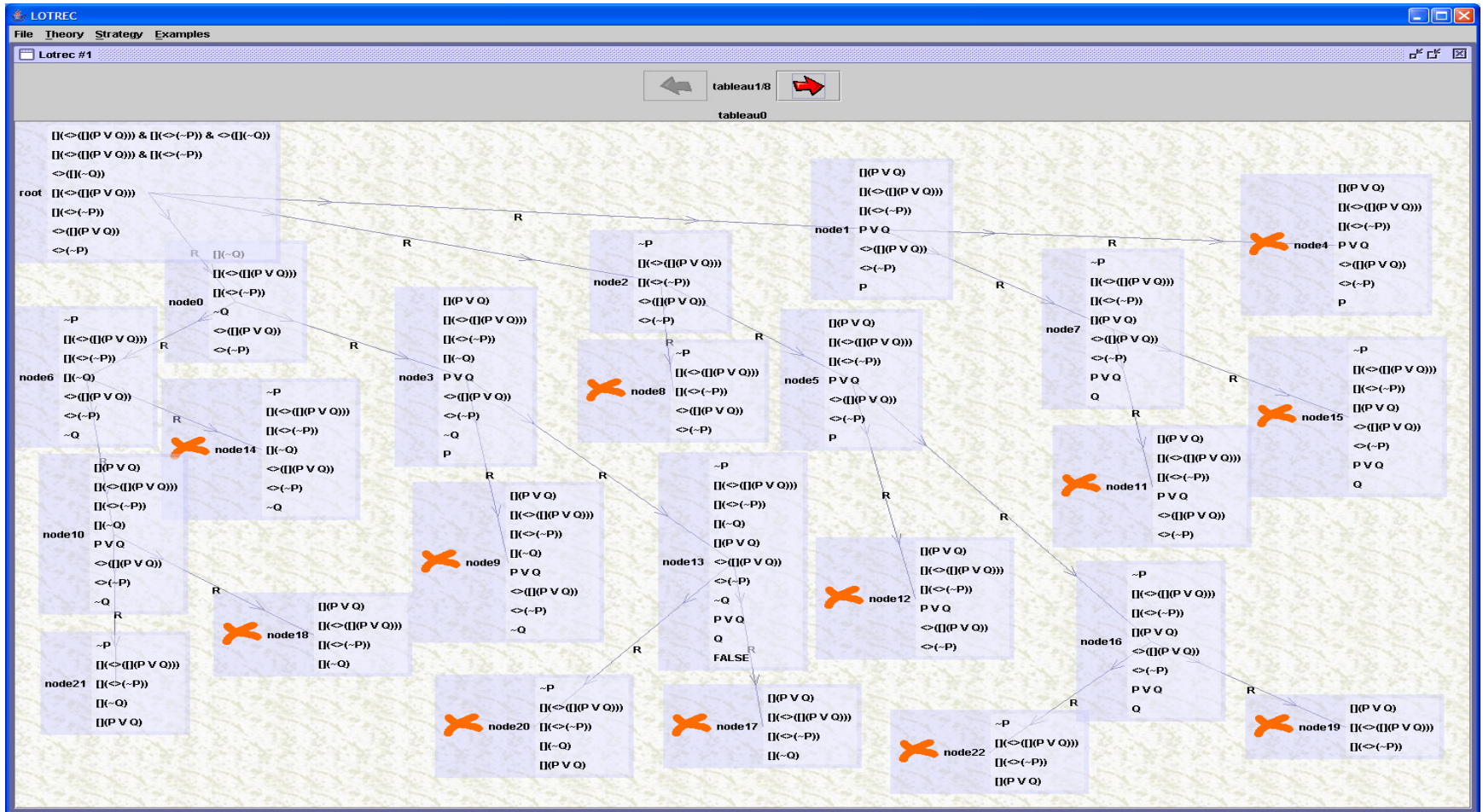
```
// rule for transitivity
rule copyNec
  if hasElement node0 nec (variable A)
  if isLinked node0 node1 R
  do add node1 nec (variable A)
end

rule createSuccessor
  if hasElement node0 not nec (variable A)
  if isNotMarked node0 CONTAINED
  do newNode node1
  do link node0 node1 R
  do add node1 not (variable A)
end

// inclusion test
rule loopTest
  if isNewNode node1
  if isAncestor node0 node1
  if contains node0 node1
  do mark node1 CONTAINED
end
```

S4Strategy

$(\Box \leftrightarrow \Box (P \vee Q)) \ \& \ \Box \leftrightarrow \sim P \ \& \ \leftrightarrow \Box \sim Q$



Overview

- tableaux systems: basic ideas
- tableaux systems: basic definitions
- tableaux for simple modal logics
- tableaux for transitive modal logics
- **tableaux for modal logics with transitive closure and other modal logics**
- some implemented tableaux theorem provers

Linear Temporal Logic

- two modal operators:
 - \Box = always
 - X = next
- R_X is serial and deterministic
- $R_{\Box} = R_X^*$
 - $R(\Box)$ linear order
- fixpoint axioms:
 - $\Box A \leftrightarrow A \wedge X\Box A$
 - $\langle \rangle A \leftrightarrow A \vee X\langle \rangle A$
- least fixpoint axiom:
 - $A \wedge \Box(A \rightarrow XA) \rightarrow \Box A$
- decidable, EXPTIME complete

Tableau rules for Linear Temporal Logic

how take induction into account?

- solution: don't care, and only apply the mix axioms:
 - rewrite $\Box A$ to $A \wedge X\Box A$
 - rewrite $\langle \rangle A$ to $A \vee X\langle \rangle A$
- only create successors for X , never for $\langle \rangle$
- termination: use the looptest from transitive modal logics
 - nodes only contain subformulas of orig. formula
 - looptest succeeds at most at polynomial depth

Tableau rules for Linear Temporal Logic: example

- Example: $w \Vdash \Box P$

add $w \Vdash P \wedge X\Box P$ (by fixpoint axioms)

add $w \Vdash P, w \Vdash X\Box P$

create u , add $R_X wu$, add $u \Vdash \Box P$

(by propagation rule for X)

add $u \Vdash P \wedge X\Box P$ (by mix axioms)

add $u \Vdash P, u \Vdash X\Box P$

w contains u : mark u “contained”

Tableau rules for Linear Temporal Logic (ctd.)



- may result in ‘nonstandard’ models of $\langle \rangle P$
 - ➔ “ P never fulfilled”
 - ➔ check if all $\langle \rangle$ are fulfilled!

Tableau rules for Linear Temporal Logic: example

- Example: LTLStrategy($\langle \rangle P$)

$w \Vdash \langle \rangle P$

Tableau rules for Linear Temporal Logic

- Example: LTLStrategy($\langle \rangle P$)

$w \Vdash \langle \rangle P$

$w \Vdash P \vee X \langle \rangle P$

(by fix)

Tableau rules for Linear Temporal Logic

- Example: LTLStrategy($\langle \rangle P$)

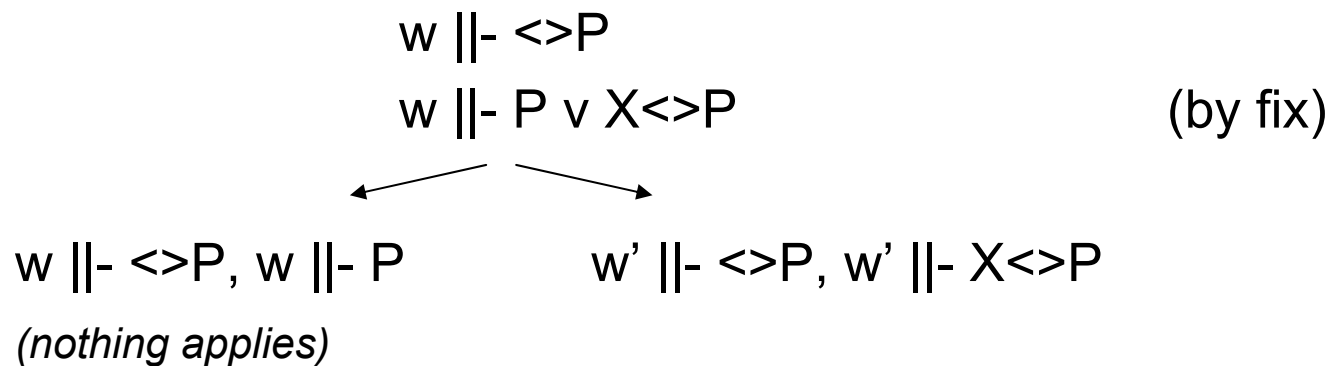


Tableau rules for Linear Temporal Logic

- Example: LTLStrategy($\langle \rangle P$)

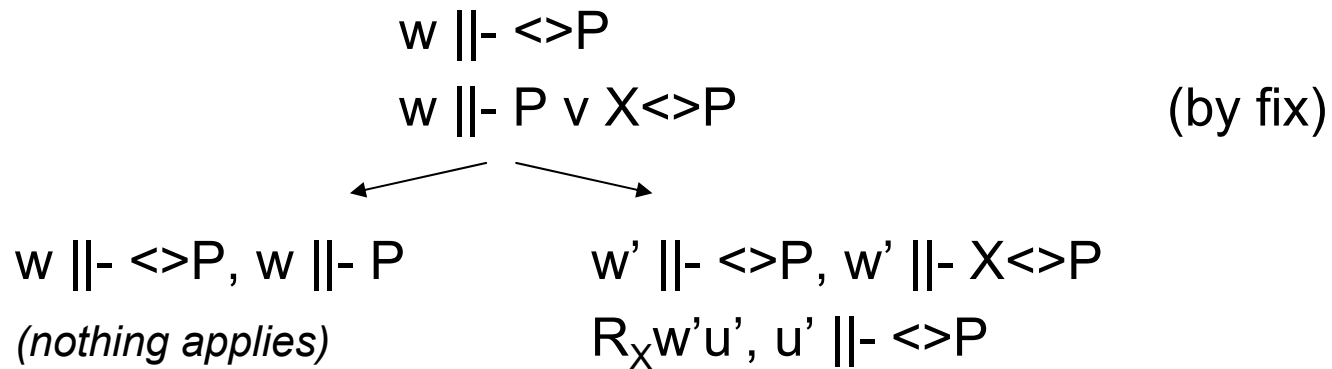


Tableau rules for Linear Temporal Logic

- Example: LTLStrategy($\langle \rangle P$)

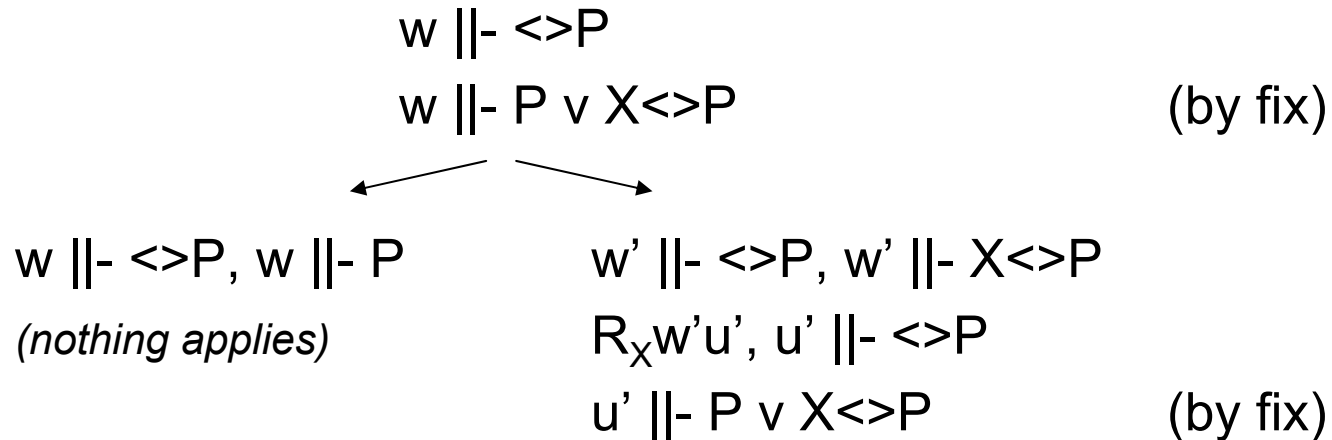


Tableau rules for Linear Temporal Logic

- Example: LTLStrategy($\langle \rangle P$)

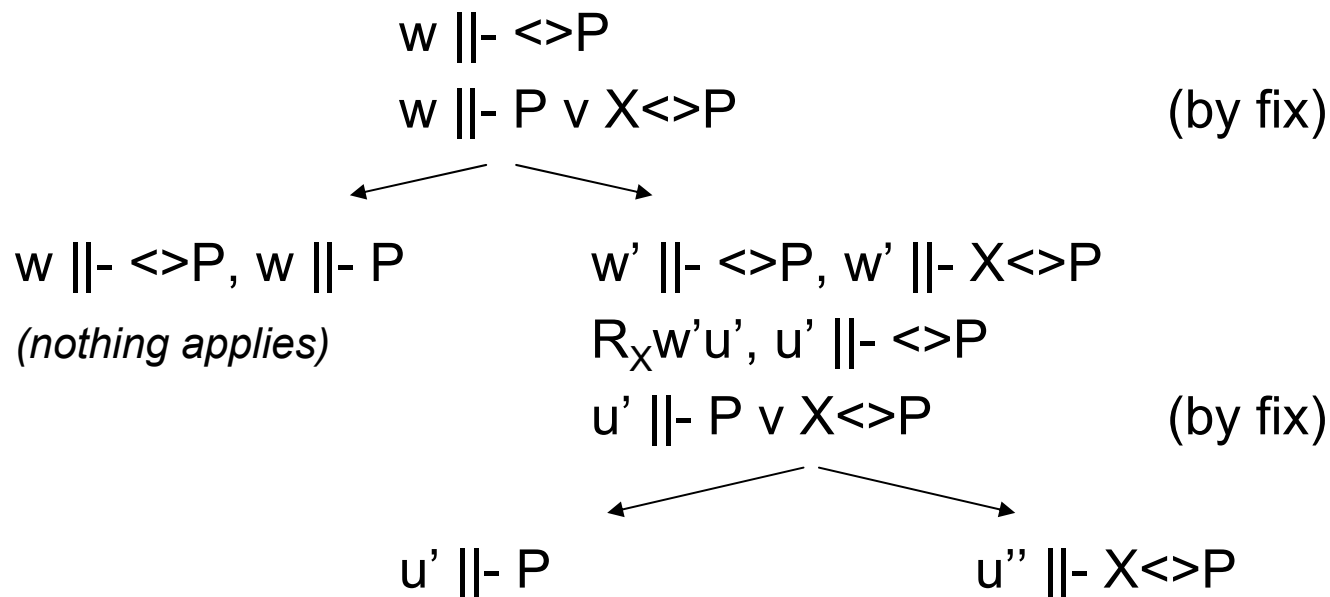


Tableau rules for Linear Temporal Logic

- Example: LTLStrategy($\langle \rangle P$)

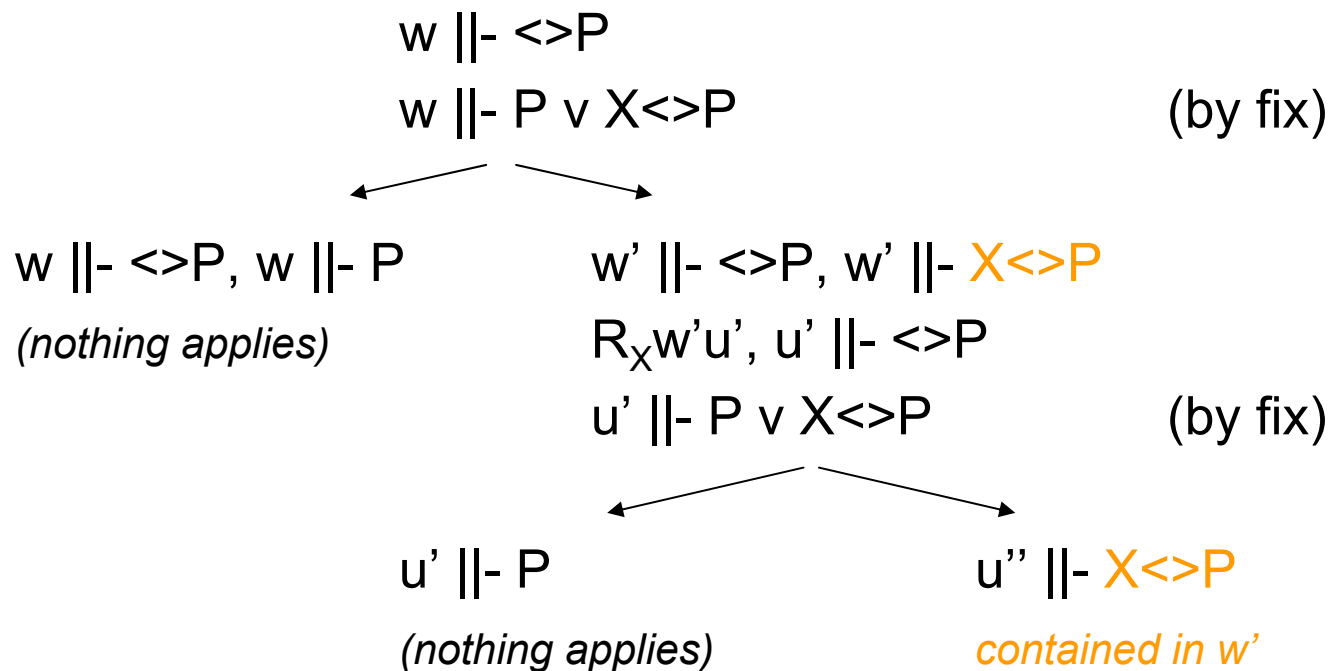
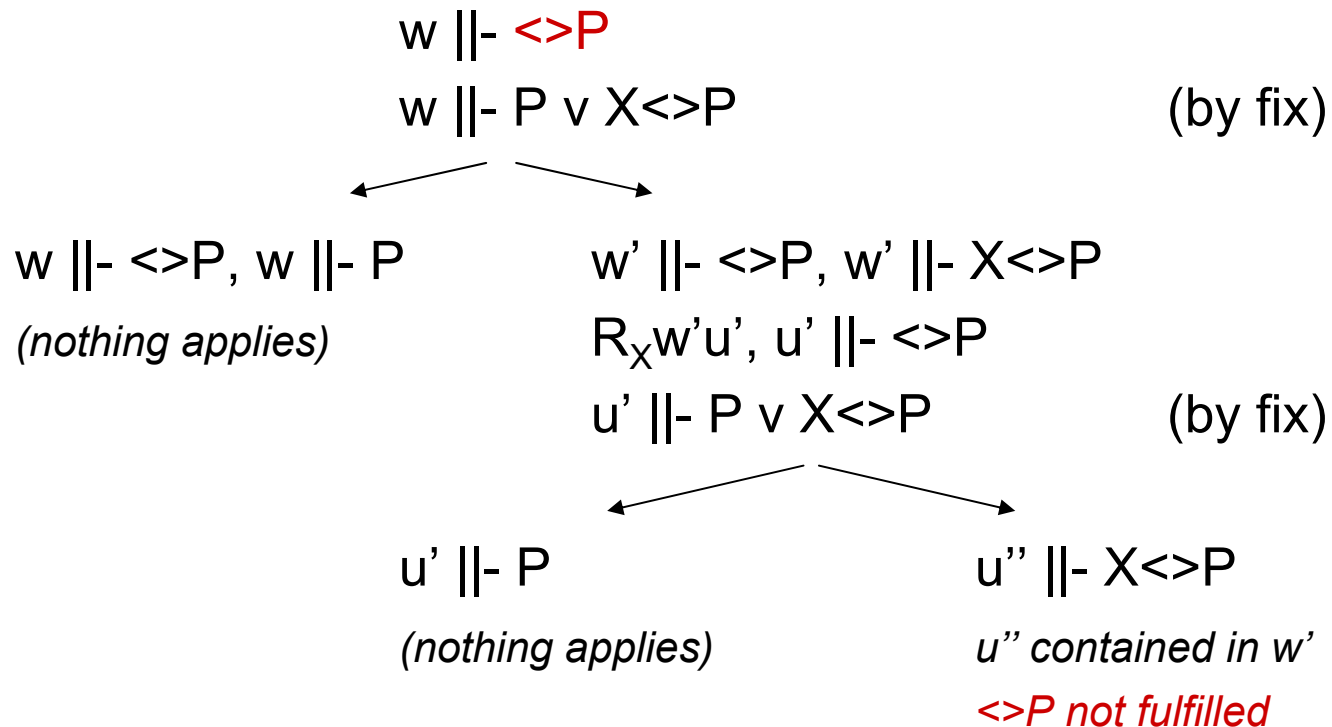


Tableau rules for Linear Temporal Logic

- Example: LTLStrategy($\langle \rangle P$)



Propositional dynamic logic (PDL)

- two kinds of expressions

- formulas:

$$A ::= P \mid \sim A \mid A \wedge B \mid [\pi]A$$

- programs:

$$\pi ::= a \mid \pi_1; \pi_2 \mid \pi_1 \cup \pi_2 \mid \pi^* \mid A?$$

- in the models: R interprets programs

$$R(\pi_1; \pi_2) = R(\pi_1); R(\pi_2)$$

$$R(\pi_1 \cup \pi_2) = R(\pi_1) \cup R(\pi_2)$$

$$R(\pi^*) = (R(\pi))^*$$

$$R(A?) = \{ \langle w, w \rangle : w \Vdash A \}$$

Tableaux for PDL

- similar to LTL:
 - expand $[\pi^*]A$ to $A \wedge [\pi][\pi^*]A$
 - don't apply createSuccessor to formulas $\sim[\pi^*]A$
 - mark nodes that are included in some ancestor
 - don't apply createSuccessor to formulas $\sim[\pi]A$ if node is marked
 - expand the other program expressions:

$[\pi_1;\pi_2]A$	\leftrightarrow	$[\pi_1][\pi_2]A$
$[\pi_1 \cup \pi_2]A$	\leftrightarrow	$[\pi_1]A \wedge [\pi_2]A$
$[A?]B$	\leftrightarrow	$A \rightarrow B$

Description logics

- “roles” and “concepts”
 - more expressive than classical propositional logic
 - less expressive than 1st order logic
- focus on decidable logics
- applications:
 - databases
 - software engineering
 - web-based information systems
 - description of medical terminology
 - ontology of the semantic web
 - standards: DAML+OIL, OWL
 - description of web services
 - WSDL, OWL-S

Description logics: concepts and roles

- roles = binary relations
 - hasChild
 - hasHusband
- concepts = unary relations = properties
 - Person
 - Female
 - Parent \cap Female
 - Father \cup Mother
 - \sim Parent
 - \exists hasChild.Female “individuals having a female child”
 - \forall hasChild.Female “...”
 - >1 hasChild.T “individuals having more than 1 child”
- set of concepts \rightarrow “assertion box” (ABox)

Description logics: TBoxes

- set of relations between concepts and roles
 - “terminological box” (TBox)
 - restricted to concept abbreviations (sometimes: fixpoint definitions)
Mother = Person \cap Female
 - are expanded away → TBox = \emptyset

Description logics: reasoning tasks

- satisfiability of a concept C
- subsumption of C_1 by C_2
same as: $C_1 \cap \sim C_2$ unsatisfiable
- equivalence of C_1 by C_2
same as: C_1 subsumes C_2 and C_1 subsumes C_2
- disjointness of C_1 and C_2
 \perp subsumes $C_1 \cap C_2$

→ all reasoning tasks reduce
to concept satisfiability

Description logics

- translation of concepts into modal logics

$\exists \text{hasChild.Female} = \langle \text{hasChild} \rangle \text{Female}$

$\forall \text{hasChild.Female} = [\text{hasChild.Female}]$

$\text{Parent} \cap \text{Female} = \text{Parent} \wedge \text{Female}$

$\text{Father} \cup \text{Mother} = \text{Father} \vee \text{Mother}$

$\langle 2 \text{ hasChild.T} = [\text{hasChild}]_2 \text{T}$

$\geq 2 \text{ hasChild.T} = \langle \text{hasChild} \rangle_2 \text{T}$

...modal logics with number restrictions

[Fattorosi&Barnaba, van der Hoek]

Description logics

- description logic ALC:

$\sim C$

$C_1 \cap C_2$

$C_1 \cup C_2$

$\exists R.C$

$\forall R.C$

= multimodal K

- description logic $ALC_{reg} =$

ALC + regular expressions on roles

= PDL

- all description logic reasoning tasks reduce to satisfiability checking in modal logics
- tableaux used as optimal decision procedures

Logics of action and knowledge

- 2 modal operators

$\text{K}_{nw_i} A$ “agent i knows that A ”

$[a] A$ “after execution of action a , A holds”

- “product logics”:

$R_{\text{K}_{nw_i}} \circ R_a = R_a \circ R_{\text{K}_{nw_i}}$ (permutation)

if $wR_{\text{K}_{nw_i}}u$ and $wR_a v$ then exists t such that $uR_a t$ and $vR_{\text{K}_{nw_i}} t$ (confluence)

- axiomatically:

$\text{K}_{nw_i}[a]A \leftrightarrow [a]\text{K}_{nw_i}A$

$\langle a \rangle \text{K}_{nw_i}A \rightarrow \text{K}_{nw_i}\langle a \rangle A$

tableaux: ...

→ problem: combination with transitivity

Belief-Desire-Intention logics

- [Bratman, Rao&Georgeff]
- 3 modal operators
 - Bel_i A “agent i believes that A”
 - Desire_i A “agent i desires that A”
 - Intend_i A “agent i intends that A”
- plus branching time logic

Modal logics with density

- accessibility relation is dense
if Rwu then exists $v : Rwv$ and Rvu
- ...

Non-normal modal logics

- no accessibility relation, but neighborhood functions: $N: W \rightarrow 2^{2^W}$
 - $w \Vdash \Box A$ iff exists U in $N(w)$ for all u in U : $u \Vdash A$non-normal modal logic EM
- can be represented by a set of relations
 - $w \Vdash \Box A$ iff exists R_i for all u ($R_i w u$ implies $u \Vdash A$)
- logic EM: “non-normal”
 - not valid: $\Box P \wedge \Box Q \rightarrow \Box(P \wedge Q)$
 - but valid: $\Box(P \wedge Q) \rightarrow \Box P \wedge \Box Q$

Tableau rules for EM

- ...

Overview

- tableaux systems: basic ideas
- tableaux systems: basic definitions
- tableaux for simple modal logics
- tableaux for transitive modal logics
- tableaux for modal logics with transitive closure and other modal and description logics
- **tableaux for 1st order logic**
- some implemented tableaux theorem provers

1st order logic

- How should we handle the quantifiers?

$\forall x p(x) \wedge \sim p(a)$ is unsatisfiable

$\forall x p(x) \wedge \exists x \sim p(x)$ is unsatisfiable

- naïve implementation [Beth, Smullyan]:

if hasElement node0 forall x A(x)

do createTerm t

(doesn't exist in LoTREC yet)

do add node0 A(t)

if hasElement node exists x A(x)

do createNewConstant c

do add node A(c)

→ problem: loops for satisfiable formulas

Herbrand Tableaux for 1st order logic

- 1st solution: restrict instantiation to Herbrand universe
if hasElement node0 forall x A(x)
do createHerbrandTerm t *(doesn't exist in LoTREC yet)*
do add node0 A(t)
- ex.: $\exists x p(x,x) \wedge \exists x \forall y \sim p(x,y)$ satisfiable
 1. $\exists x p(x,x)$
 2. $\exists x \forall y \sim p(x,y)$
 3. $\forall y \sim p(a,y)$ (2), new constant
 4. $\sim p(a,a)$ (3), only Herbrand term
 5. $p(b,b)$ (1), new constant
 6. $\sim p(a,b)$ (3), Herbrand termno further instantiation of (3) is possible
- decision procedure for formulas without positive $\forall \dots \exists$

Herbrand Tableaux for 1st order logic

- counterexample: $\forall x \exists y p(x,y)$ satisfiable

1. $\forall x \exists y p(x,y)$

2. $\exists y p(a,y)$

(1), Herbrand term

3. $p(a,b)$

(2), new constant

4. $\exists y p(b,y)$

(1), Herbrand term

5. $p(b,c)$

(4), new constant

6. ...

→ loops

Free-variable tableaux with unification

- 2nd solution: don't instantiate at all
 - work with free variables
 - runtime skolemization of existential quantifiers
 - term unification
- ex.: $\forall x \exists y p(x,y) \wedge \forall x \exists y \sim p(x,y)$ satisfiable
 1. $\forall x \exists y p(x,y)$
 2. $\forall x \exists y \sim p(x,y)$
 3. $\exists y p(x_1,y)$ from (1), replace x by free x_1
 4. $\exists y \sim p(x_2,y)$ from (2), replace x by free x_2
 5. $p(x_1, f(x_1))$ from (3), Skolem function $f(x_1)$
 6. $\sim p(x_2, g(x_2))$ from (4), Skolem function $g(x_2)$

stops: (5) and (6) don't unify
- ... but does not terminate in all cases (sure)
 - else 1st order logic would be decidable

Overview

- possible worlds semantics: quickstart
- tableaux systems: basic ideas
- tableaux systems: basic definitions
- tableaux for simple modal logics
- tableaux for transitive modal logics
- tableaux for intuitionistic logic
- tableaux for other nonclassical logics
- tableaux for modal logics with transitive closure and other modal and description logics
- tableaux for 1st order logic
- **some implemented tableaux theorem provers**

LoTREC

- IRIT-CNRS Toulouse (Sahade, Gasquet, Herzig); accessible through [www](http://www.lo-trec.fr)
- general theorem prover
- explicit accessibility relations
- easy to implement logics with symmetric accessibility relations etc.
 - back-and-forth rules
- inefficient

TableauxWorkBench (TWB)

- Australian National U. (Abate, Goré)
- general theorem prover
- close to Gentzen sequents
- accessibility relations remain implicit
- hard to implement logics with symmetric accessibility relations
 - temporal logic with future and past
 - converse of programs

LogicWorkBench (LWB)

- U. Bern (Jäger, Heuerding); accessible through [www](#)
- efficient algorithms for all the basic modal and temporal logics
- hard to implement a new logic

FaCT

- U. Manchester (Horrocks); open source
- fast decision procedure for description logics with inverse roles and qualified number restrictions
 - = multimodal K + converse + number restrictions
- optimized backtracking: “backjumping”

KSAT

- U. Trento (Giunchiglia, Sebastiani)
- combines tableaux method with fast SAT solvers for classical propositional logic
 - call a SAT solver, where subformulas $\Box A$, $\langle \rangle B$ are viewed as atomic
 - SAT solver returns a tentative valuation
 - use modal tableau rules to generate children
 - if inconsistent then there is no model
 - else iterate
- very efficient
- exists for all basic modal logics

KSAT (ctd.)

- $\text{KSAT}(\Box(P\&Q) \ \& \ \langle \rangle \sim P)$
 - call SAT with set of clauses $\{\Box(P\&Q), \langle \rangle \sim P\}$
 - SAT returns:
 - $V(\Box(P\&Q)) = 1$
 - $V(\langle \rangle \sim P) = 1$
 - apply createOneSuccessor and propagateNec:
 - $w \Vdash \Box(P\&Q), w \Vdash \langle \rangle \sim P, R w u, u \Vdash \sim P, u \Vdash P\&Q$
 - call SAT with set of clauses $\{P, Q, \sim P\}$
 - SAT returns:
 - set of clauses unsatisfiable*
 - $\Box(P\&Q) \ \& \ \langle \rangle \sim P$ is unsatisfiable in K

Conclusion

- search for models = exploit the truth conditions
- tableaux work both ways:
 - finding a model
 - refuting
- termination = decidability
- tableaux as optimal decision procedures
 - description logics