Tableaux Systems

Tutorial at 1st School on Universal Logic
Montreux, 26-27 March 2005

Andreas Herzig

IRIT-CNRS
Toulouse
http://www.irit.fr/~Andreas.Herzig/

What this tutorial is about

- in focus
  - the tableaux method
  - … for logics with possible worlds semantics
  - … and combinations thereof
  - … as a computerized proof system (LoTREC)
- not in focus:
  - tableau**s**
  - proof theory, sequent calculi (cf. course on LDS)
  - completeness proofs
  - efficiency issues

Overview

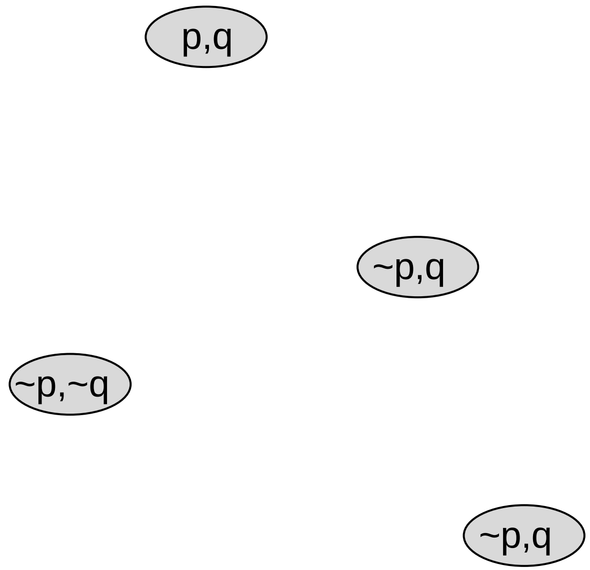
- possible worlds semantics: quickstart
- tableaux systems: basic ideas
- tableaux systems: basic definitions
- tableaux for simple modal logics
- tableaux for transitive modal logics
- tableaux for intuitionistic logic
- tableaux for other nonclassical logics
- tableaux for modal logics with transitive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

Possible worlds

- possible world ➔ valuation of classical logic

  w ||- P iff
  $V_w(P) = 1$, for P in Atoms

  w ||- A∧B iff
  (w ||- A and w ||- B)

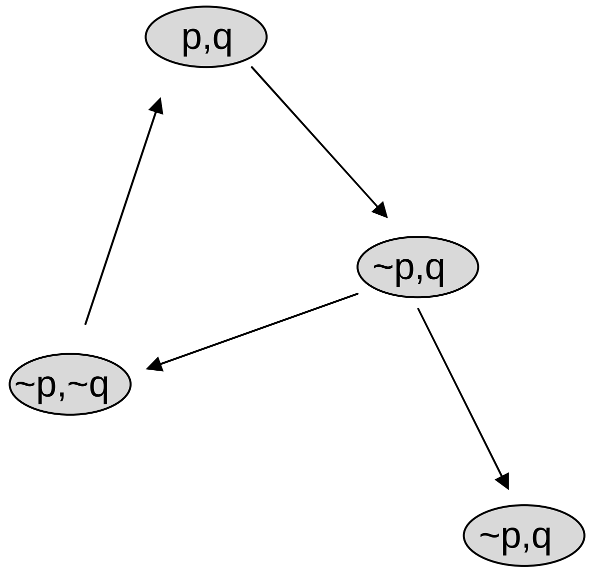


Possible worlds models

- possible worlds model
  = labeled graph
  = transition system

- node = possible world
  - valuation of classical logic
  - not every valuation appears (some logically possible worlds are not actually possible)
  - $V_w = V_u$ does not imply $w = u$

- link = accessibility relation R



Possible worlds models: accessibility relations

- temporal

  Rwu iff u is in the future of w
- alethic

  Rwu iff u is possible, given the actual world w
- epistemic

  $R_i$wu iff u is possible for agent i, given the actual world w
- deontic

  Rwu iff u is an ideal version of w
- dynamic

  $R_a$wu iff u is a possible result of the execution of program/action a in w
- comparative (preferential, …)

  Rwu iff w is smaller than u

  $R_v$wu iff w is smaller than u, given v

reading of R ➔ properties of R

Possible worlds models: properties of R

- monomodal
  - serial: forall w exists u Rwu
  - reflexive
  - transitive
  - Euclidian
  - confluent (Church-Rosser)
  - dense
  - …
  - well-founded (not FO-definable!)
  - …
- multimodal
  - R_1 included in R_2
  - $R_1 = R_2 \cup R_3$
  - $R_2 = (R_1)^{-1}$ (transitive closure)
  - $R_2 = (R_1)^*$ (transitive closure)
  - $R_1 \circ R_2 = R_2 \circ R_1$
  - Church-Rosser
  - …

Language: modal operators

- express intensional concepts (belief, time, action, obligation, …)
- non truth functional
- schema: op(a_1,…,a_n), where op is the name of the operator, and a_i some argument
- generic form:
  - []A = A is necessary (true in all possible worlds)
  - <>A = A is possible
- in general: []A same as ~<>~A
  - except in substructural logics (intuitionistic, …)

Language: modal operators

- temporal
  - []A = henceforth A (true in all future time points)
  - <>A = eventually A
- deontic
  - []A = A is obligatory (true in all ideal worlds)
  - <>A = A is permitted (~<>A = A is forbidden)
- epistemic
  - $[]_i$A = i believes A (true in all worlds possible for i)
  - $<>_i$A = ..
- dynamic
  - [a]A = A is true after (every possible way of) executing a
  - <a>A = …
- conditional
  - A => B = if A then B
proof of A can be transformed into proof of B (intuitionistic)
if A was true then B would be true (counterfactual)

Interpreting the language: truth conditions

- classical connectives

  w ||- P   iff V_w(P) = 1, for P in Atoms

  w ||- A$\wedge$B   iff (w ||- A and w ||- B)

- interpretation of non-classical connectives
  - via accessibility relation R
- schema:

  w ||- op(a_1,…,a_n) iff Cond(op,a_1,…,a_n,w,R)

- the basic modal operators:

  w ||- []A   iff forall u: Rwu implies u ||- A

  w ||- <>A   iff exists u: Rwu and u ||- A

Examples of truth conditions

- multimodal operators

  w ||- $[]_i A$ iff forall u: $R_i wu$ implies u ||- A

  w ||- $<>_i A$ iff …

- relation algebra operators

  w ||- $[]^{-1} A$ iff forall u: $R^{-1} wu$ implies u ||- A

  w ||- $[]_{i \cup j} A$ iff forall u: $(R_i \cup R_j) wu$ implies u ||- A

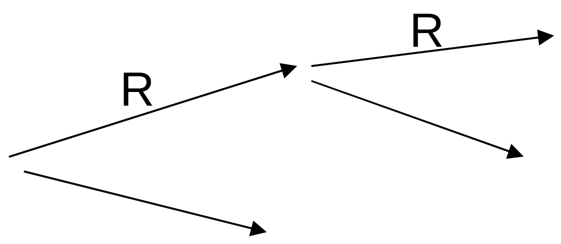
  w ||- $[]^* A$ iff forall u: $R^* wu$ implies u ||- A)

- non-normal operators

  w ||- <>A iff forall R_i exists u: $R_i wu$ and u ||- A

  w ||- []A iff exists R_i forall u …

Examples of truth conditions: temporal operators

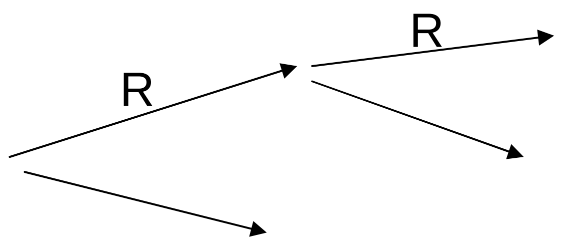


- branching time operators

w ||- ∃XA iff ∃R in Paths(w): R(w) ||- A

(Paths(w) = the set of paths going through w)

Examples of truth conditions: temporal operators



- branching time operators

  w ||- ∃XA iff ∃R in Paths(w): R(w) ||- A

  (Paths(w) = the set of paths going through w)

  w ||- ∀<>A iff ∀R in Paths(w) ∃n R^n(w) ||- A

Examples of truth conditions: temporal operators

- binary temporal operators

  w ||- A Until B iff exists u: $R^{*}wu$ and u ||- B and
  forall u' ($R^{*}wu'$ and $R^{+}vu'$ implies u' ||- A)

  w ||- A Since B iff …

  w ||- ∀(A Until B) iff forall R in Paths(w) …

Examples of truth conditions: implications

- intuitionistic implication

  w ||- A => B iff forall u: Rwu implies u ||- A $\rightarrow$ B

- conditional operator

  w ||- A => B iff forall u: $R_{[A]}$wu implies u ||- B

- relevant implication

  w ||- A => B iff forall u,u':

  Rwuu' implies (u ||- A implies u' ||- B)

Models

- model M = (W,R,V)
  - W nonempty set (possible worlds)
  - R: Ops $\rightarrow$ (WxW) (accessibility relation)
  - V: W $\rightarrow$ (Atoms $\rightarrow$ {0,1}) (valuation)
- pointed model ((W,R,V),w)
  - w in W (actual world)
- extension of A in M

  $[A]_M$ = {w in W : w ||- A}

Validity and satisfiability

- K = the set of all models (Kripke)
- A is *valid* in K iff $[A]_M$ = W for all M in K (|$=_K$ A)

examples: [](P v ~P)

[](P∧Q) → []P∧[]Q

[]P∧[]Q → [](P∧Q)

- A is *satisfiable* in K iff $[A]_M$ nonempy for some M in K

examples: P

P∧~[]P

P∧[]~P

[]P∧~[][]P

Validity and satisfiability in a class of models C

- Cls some subset of K

- A is *valid* in Cls iff $[A]_M = W$ for all M in Cls $(|=_{Cls} A)$

examples:

[]P $\rightarrow$ P invalid in K
[]P $\rightarrow$ P valid in the class of reflexive models
<>P $\rightarrow$ <><>P valid in transitive models

- A is *satisfiable* in Cls iff $[A]_M$ nonempy for some M in Cls

examples:

P$\wedge$~[]P satisfiable in K
P$\wedge$~[]P unsatisfiable in reflexive models

A is valid in Cls iff ~A is unsatisfiable in Cls

Classes of models: examples

- {M: card(W) = 1}

  $|=_{Cls} <>A \rightarrow []A$

- {M: card(W) = 2}

  $|=_{Cls} <>(A \wedge B) \wedge <>(\sim A \wedge B) \rightarrow []B$

- {M: card(W) finite}

  …

- {M: R([]) reflexive} = KT

  $|=_{KT} []A \rightarrow A$

- {M: R([]) transitive} = K4

  $|=_{K4} <><>A \rightarrow <>A$

- {M: R([]) equivalence relation} = S5

  $|=_{S5} A \rightarrow []<>A$

Reasoning problems

- model checking

  given A, M and w, do we have w ||- A?

- validity

  given A and Cls, is A valid in Cls?

- satisfiability

  given A and Cls, does there exist M in Cls and w in M such that w ||- A?

How can we solve them automatically?

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 idea for classical logic [Beth]

- try to find M and w by applying the truth conditions ("tableau rules")

  w ||- A∧B ➔ add w ||- A, and add w ||- B

  w ||- A v B ➔ add either w ||- A, or add w ||- B (nondet.)

  w ||- ~A ➔ "don't add w ||- A" ???

  - w ||- ~~A ➔ add w ||- A
  - w ||- ~(A v B) ➔ add w ||- ~A, and add w ||- ~B
  - w ||- ~(A∧B) ➔ add either w ||- ~A, or add w ||- ~B

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

  NO if both w ||- P and w ||- ~P ("tableau is closed")

  ELSE: for every w, if w ||- 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 ||- $P \wedge \sim(P \wedge Q)$
    2. w ||- $P \wedge \sim(P \wedge Q)$, w ||- P, w ||- $\sim(P \wedge Q)$
    3. w ||- $P \wedge \sim(P \wedge Q)$, w ||- P, w ||- $\sim(P \wedge Q)$, w ||- ~P (nondet.)
- no more truth condition applies
- can’t be a model:

  both w ||- P and w ||- ~P
- backtrack on nondeterministic choices

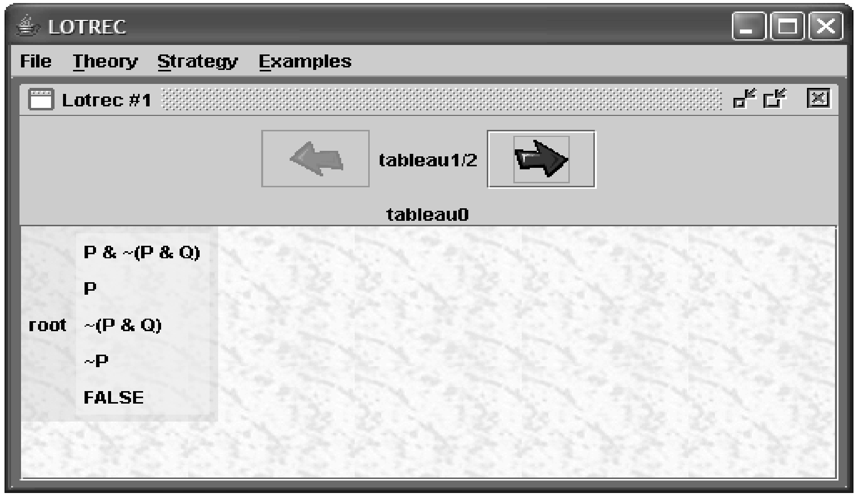
The basic idea: example for classical logic (ctd.)

- 1st downward saturated graph for

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

  ➔ not a model

  (contains P and ~P!)



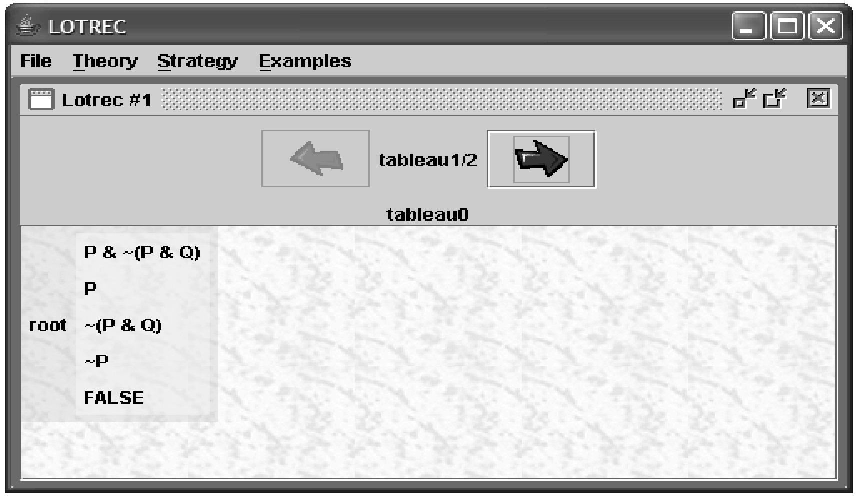
The basic idea: example for classical logic (ctd.)

- 1st downward saturated set for

  A = P ∧ ~(P∧Q)

  ➔ not a model

  (contains P and ~P!)



- 2nd downward saturated set for

  A = P ∧ ~(P∧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 ||- []A → forall u such that Rwu, add u ||- A

  w ||- <>A → add some new u, add Rwu, add u ||- A

  w ||- ~[]A → add some new u, add Rwu, add u ||- ~A

  w ||- ~<>A → …
- “downwards saturated graph”: is this a model?

The basic idea: example for modal logic

$A = P \wedge \sim[]P$

- applying tableau rules:
  1. w ||- P∧~[]P
  2. w ||- P∧~[]P, w ||- P, w ||- ~[]P
  3. w ||- P∧~[]P, w ||- P, w ||- ~[]P, Rwu, u ||- ~P

  no more tableau rule applies

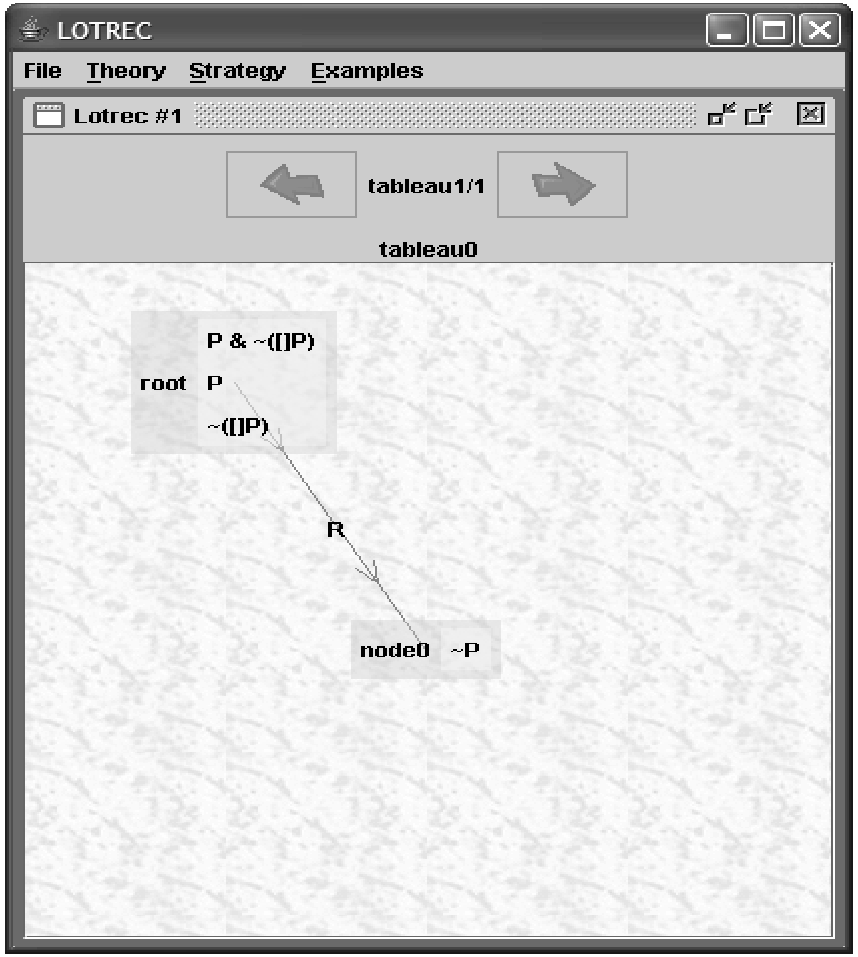
  ➔ never both w ||- A and w ||- ~A ("open tableau")
- model can be built: M = (W,R,V)

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

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

- premodel for
  A = P ∧ ~[]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 v P2 v P3) $\wedge$ (P1 v P2 v ~P3) $\wedge$ (P1 v ~P2 v P3) $\wedge$ …

  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
- rule(G) = $\{G_1,\ldots,G_n\}$

Informal definition of tableau rules

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

- application of a rule to G =
  application to *every* formula in *every* node of G.
- $\text{rule}(\{G_1,\ldots,G_n\}) = \text{rule}(G_1) \cup \ldots \cup \text{rule}(G_n)$

Tableau rules: syntax

- general form:

  rule *ruleName*

  if $cond_1$

  …

  if $cond_n$

  do $action_1$

  …

  do $action_k$

- example conditions:

  if hasElement node formula

  if isLinked $node_1$ $node_2$ R

  ... (more to come)

- example actions:

  do stop

  do addElement node formula

  do newNode node

  do link $node_1$ $node_2$ R

  do duplicate $node_1$ […]

  ... (more to come)

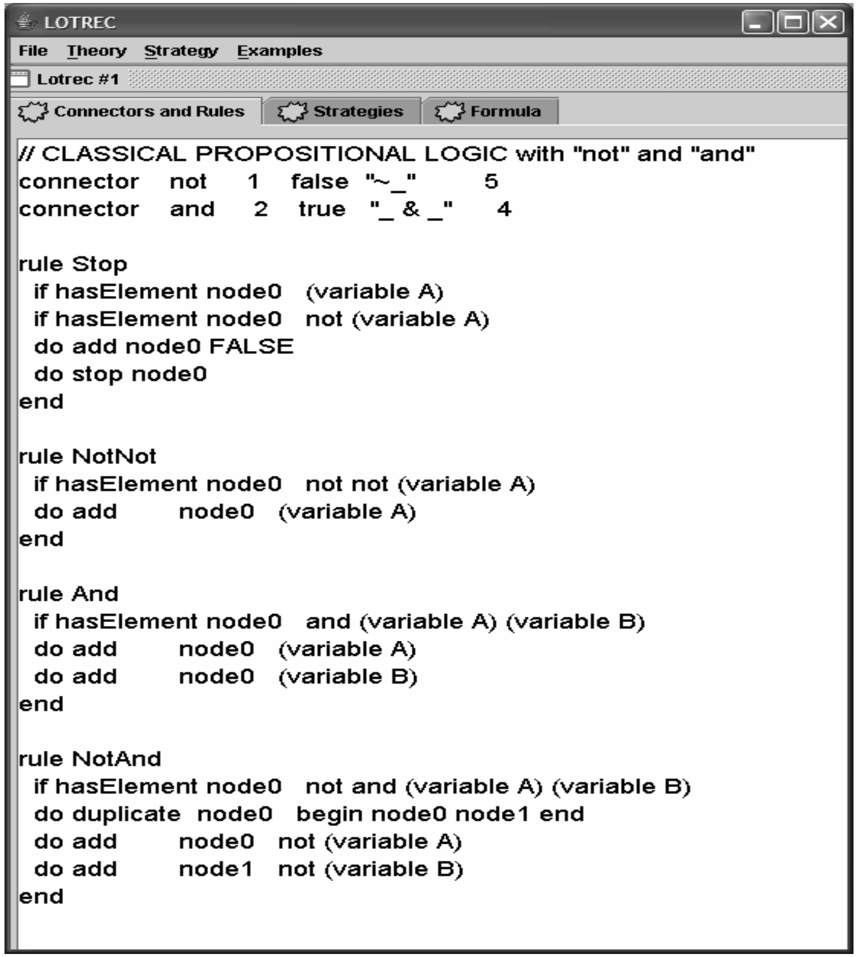
Example: tableau rules for classical logic

the

LoTREC

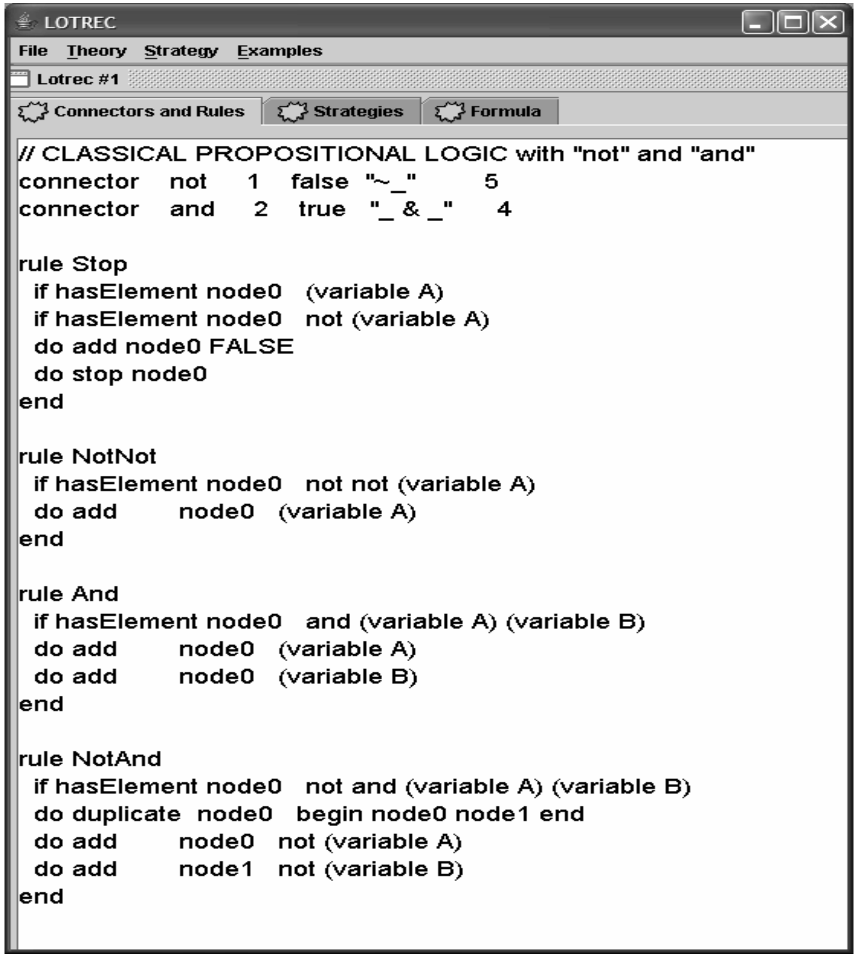
tableau

prover



Example: tableau rules for classical logic

declaration of connectors:
negation and conjunction only

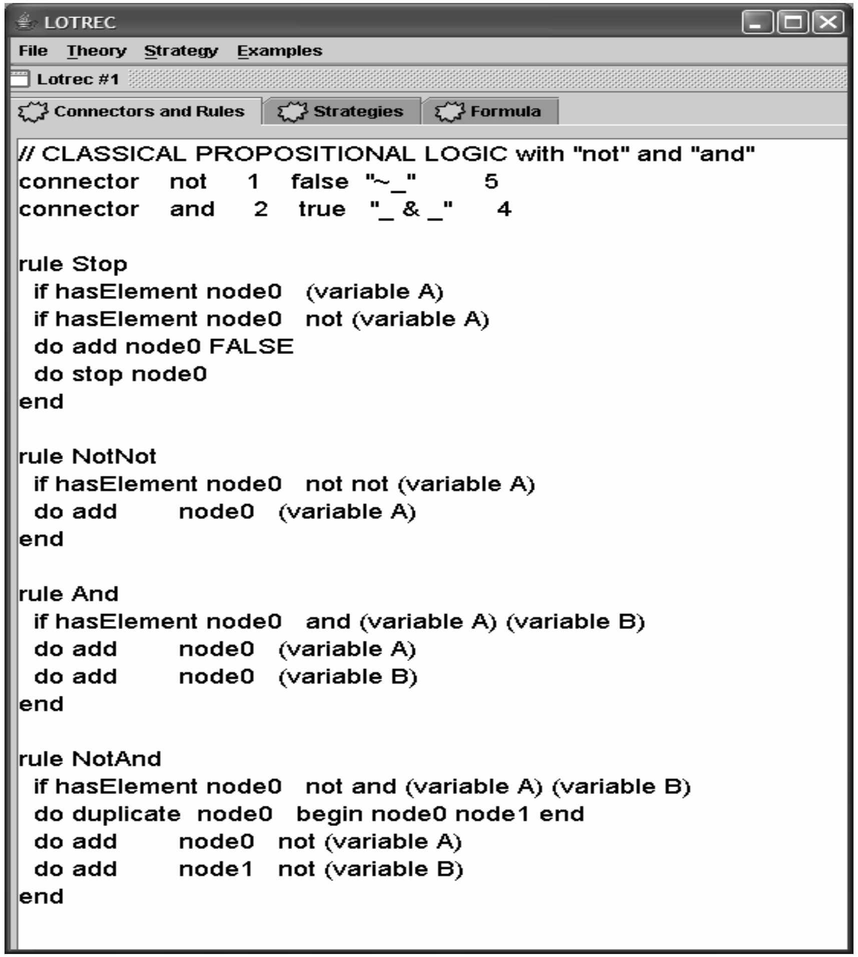


Example: tableau rules for classical logic

rule Stop:

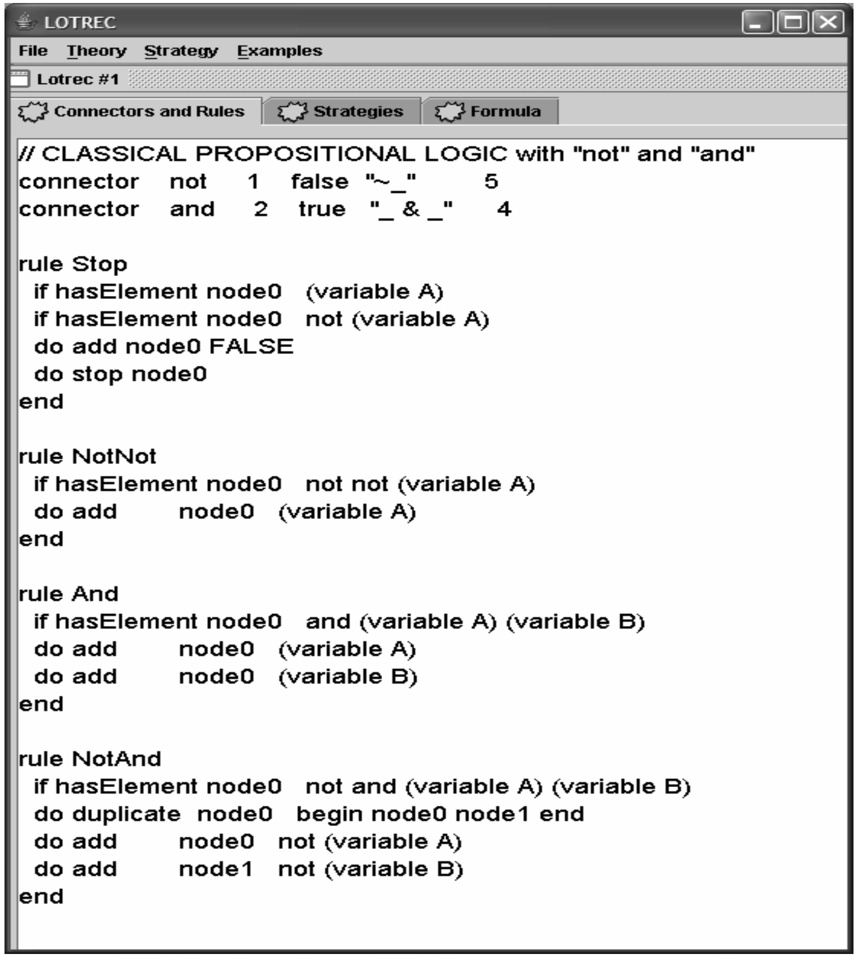
if there is an explicit contradiction

then stop exploring the tableau



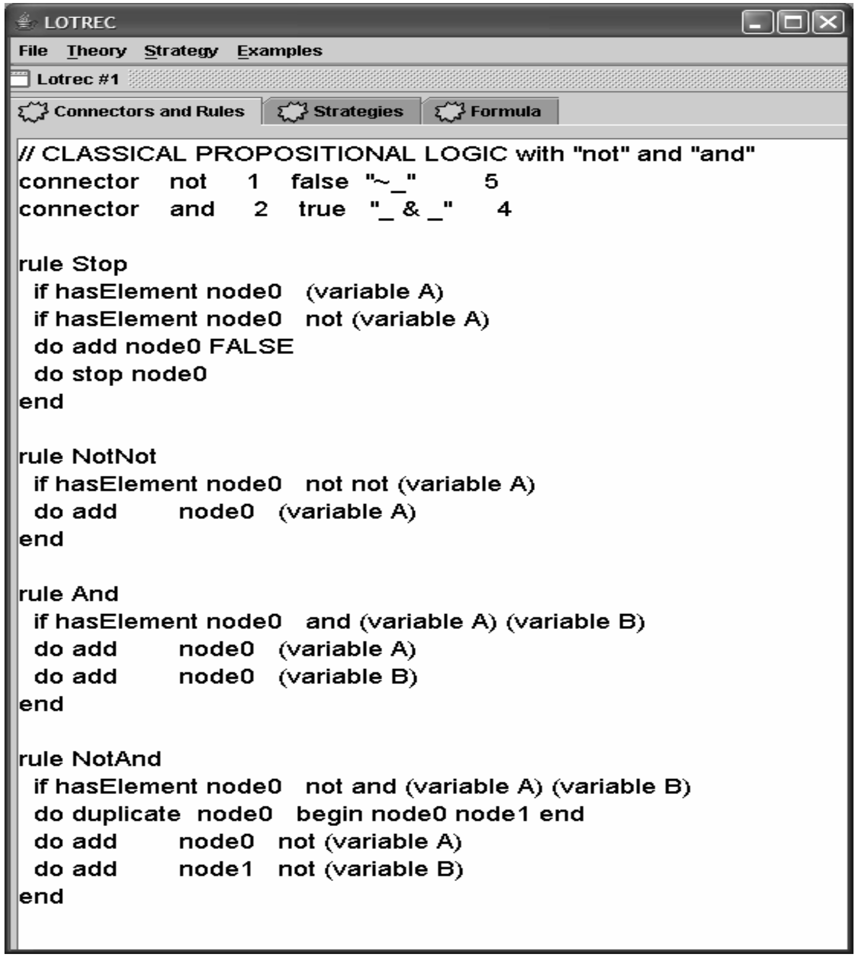
Example: tableau rules for classical logic

rule NotNot:
replaces ~~A by A



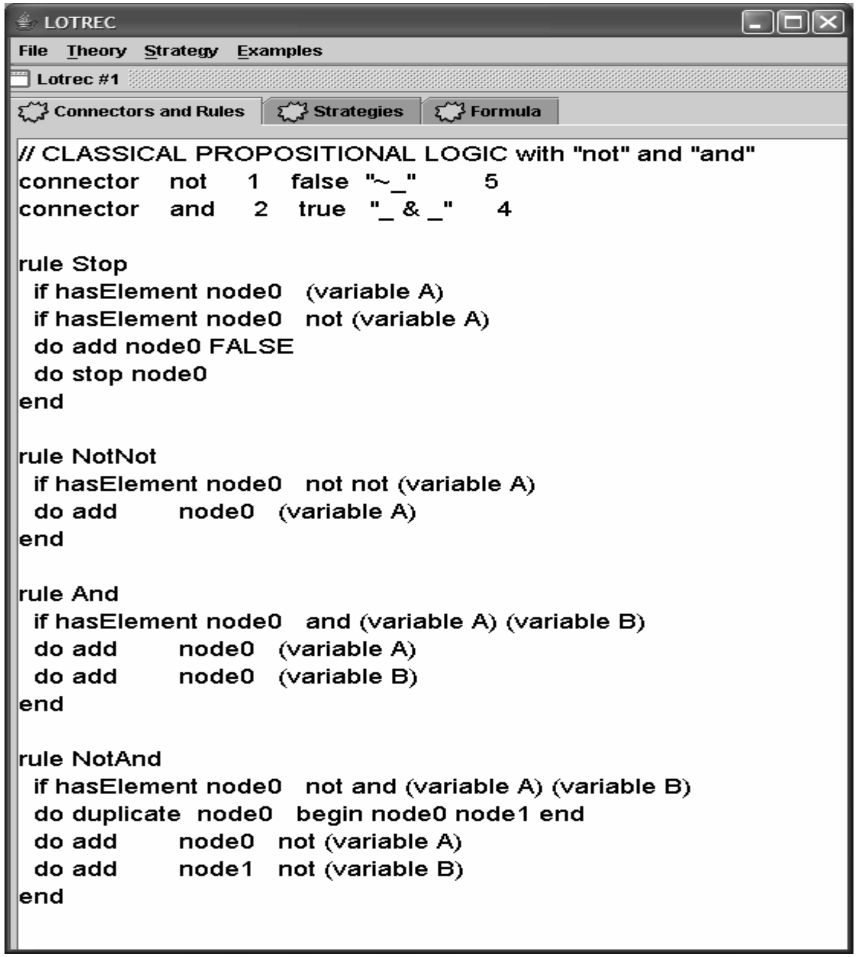
Example: tableau rules for classical logic

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



Example: tableau rules for classical logic

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



Definition of strategies

- A *strategy* defines some order of application of the tableau rules:

  firstrule $rule_1$ … $rule_n$ end

  “apply first applicable rule and stop”

  allrules $rule_1$ … $rule_n$ end

  “apply all applicable rules in order”

  repeat *strategy* end

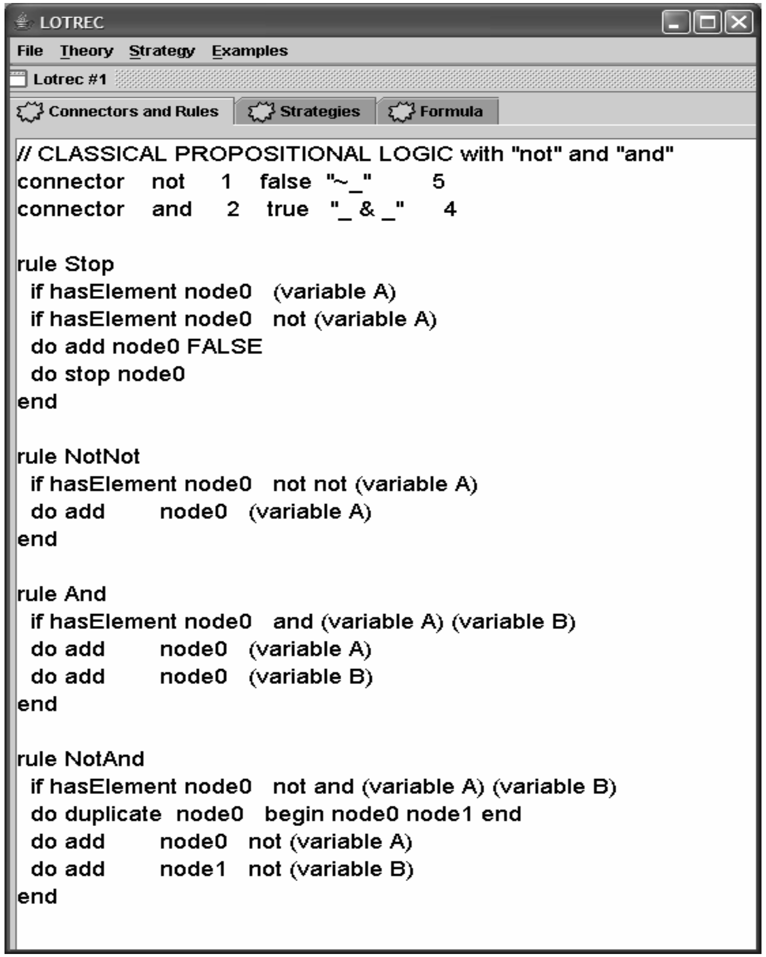
  “repeat until no rule applicable”

- Strategy stops if no rule is applicable.

Strategy
for classical logic

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

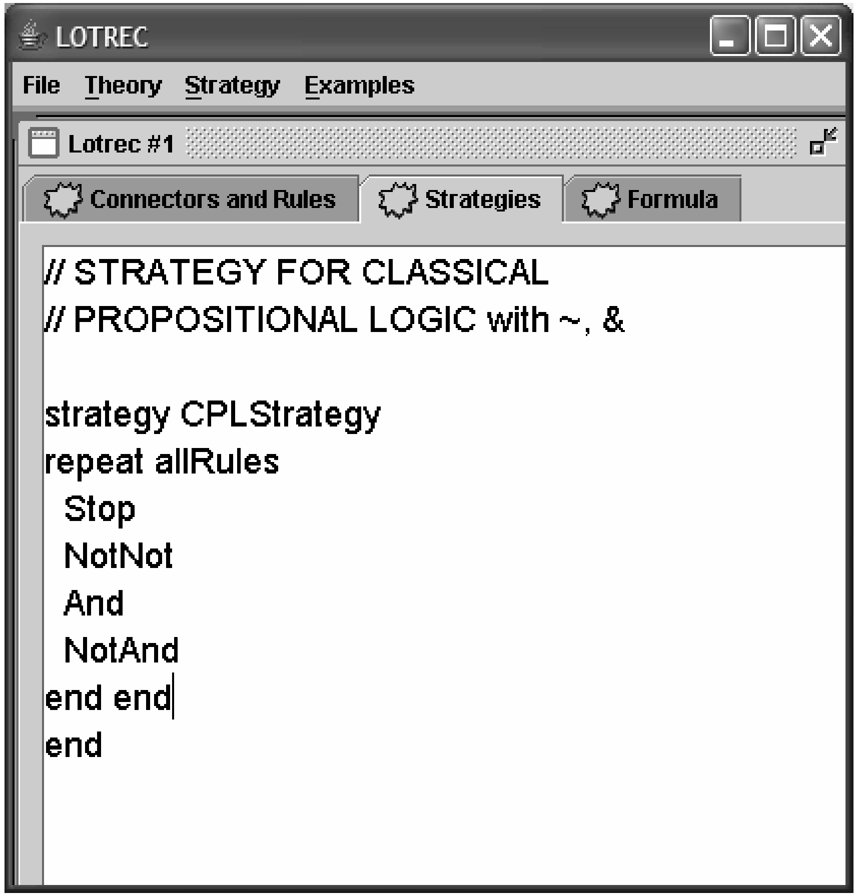
➔ "fair strategy"

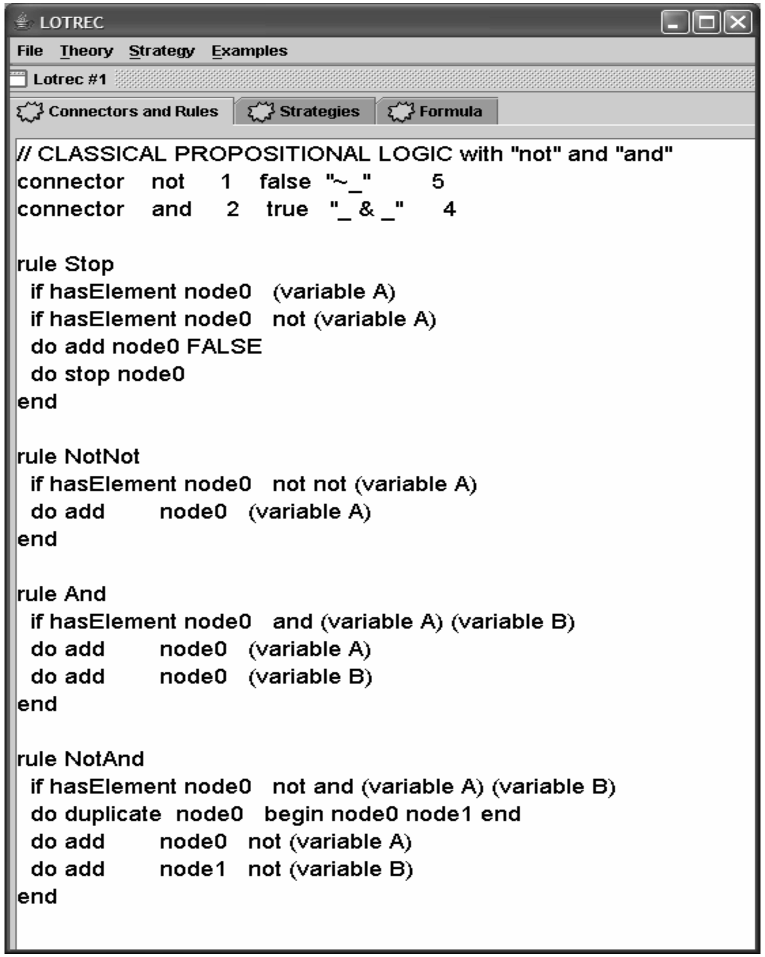



```
// CLASSICAL PROPOSITIONAL LOGIC with "not" and "and"
connector   not    1   false  "~_"      5
connector   and    2   true   "_ & _"    4

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))

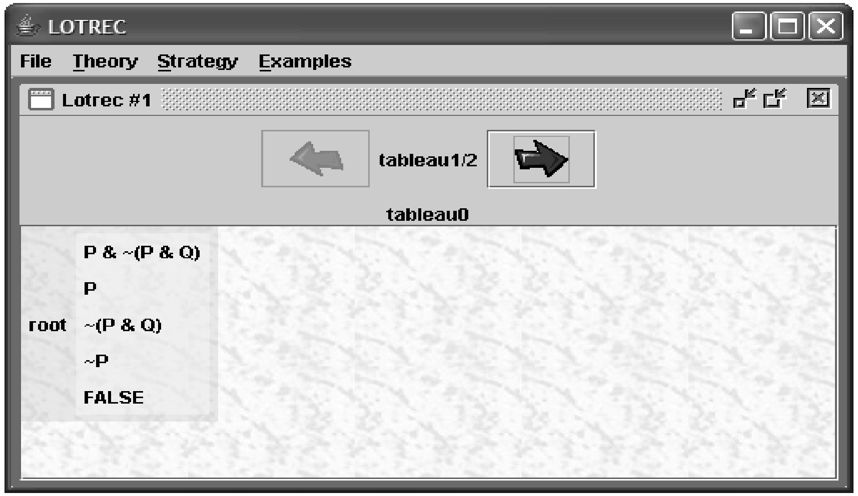




Strategy for classical logic: example (ctd.)

CPLStrategy(P&~(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, 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 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 rule(G) that is CPL-satisfiable
  - Hence if every graph is closed
    then the original A cannot be satisfiable.

Completeness

- If CPLTableaux(A) is open then A is satisfiable.
- Proof:
  - Take some open tableau G in CPLTableaux(A).

Completeness

- If CPLTableaux(A) is open then A is satisfiable.
- Proof:
  - Take some open tableau G in CPLTableaux(A).
  - G is a downwards closed set (“Hintikka set”):

    if ~~A in node then A in node

    if A&B in node then A in node and B in node

    if ~(A&B) in node then ~A in node or ~B in node

    (because allRules strategy is fair: every rule eventually applies)

Completeness

- If CPLTableaux(A) is open then A is satisfiable.
- Proof:
  - Take some open tableau G in CPLTableaux(A).
  - G is a downwards closed set (“Hintikka set”):

    if ~~A in node then A in node

    if A&B in node then A in node and B in node

    if ~(A&B) in node then ~A in node or ~B in node

    (because allRules strategy is fair: every rule eventually applies)
  - Build a CPL model from G:

    $V_{node}(P) = 1$ iff P appears in node

Completeness

- If CPLTableaux(A) is open then A is satisfiable.
- Proof:
  - Take some open tableau G in CPLTableaux(A).
  - G is a downwards closed set ("Hintikka set"):

    if ~~A in node then A in node

    if A&B in node then A in node and B in node

    if ~(A&B) in node then ~A in node or ~B in node

    (because allRules strategy is fair: every rule eventually applies)
  - Build a CPL model from G:

    $V_{node}(P) = 1$ iff P appears in node
  - Prove by induction on the form of A:

    for every A in node, $V_{node}(A) = 1$

    ("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 ||- 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 ||- []A iff forall u: Rwu implies u ||- A

  w ||- <>A iff exists u: Rwu and u ||- 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

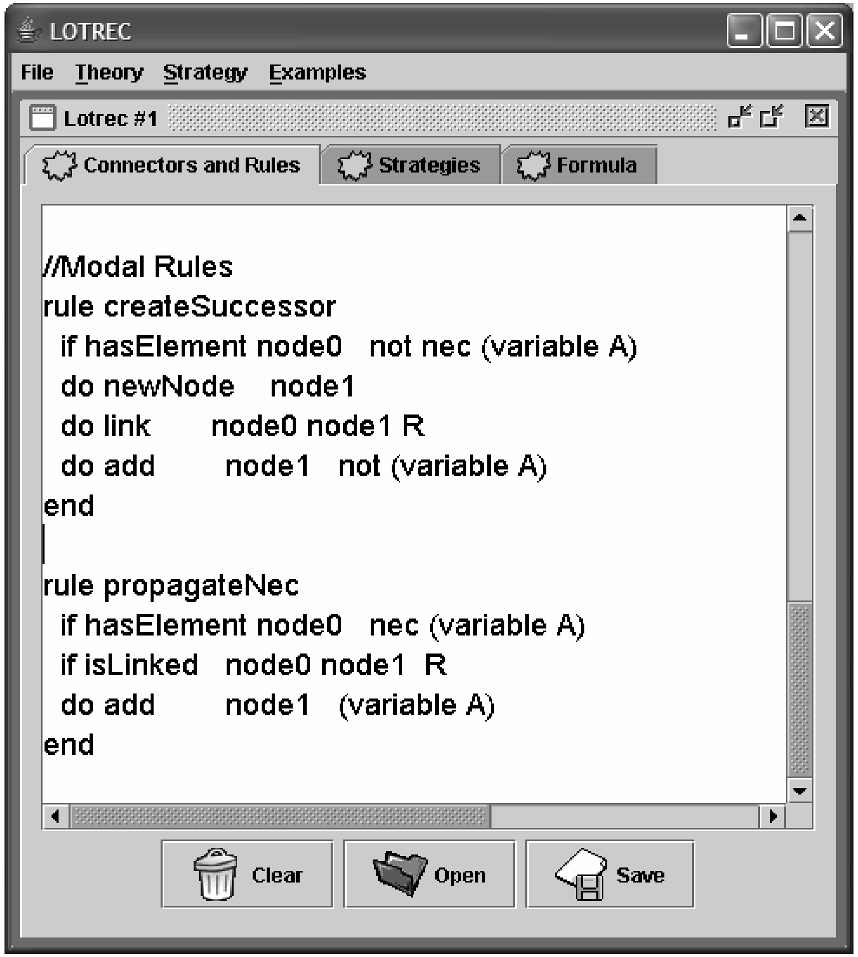
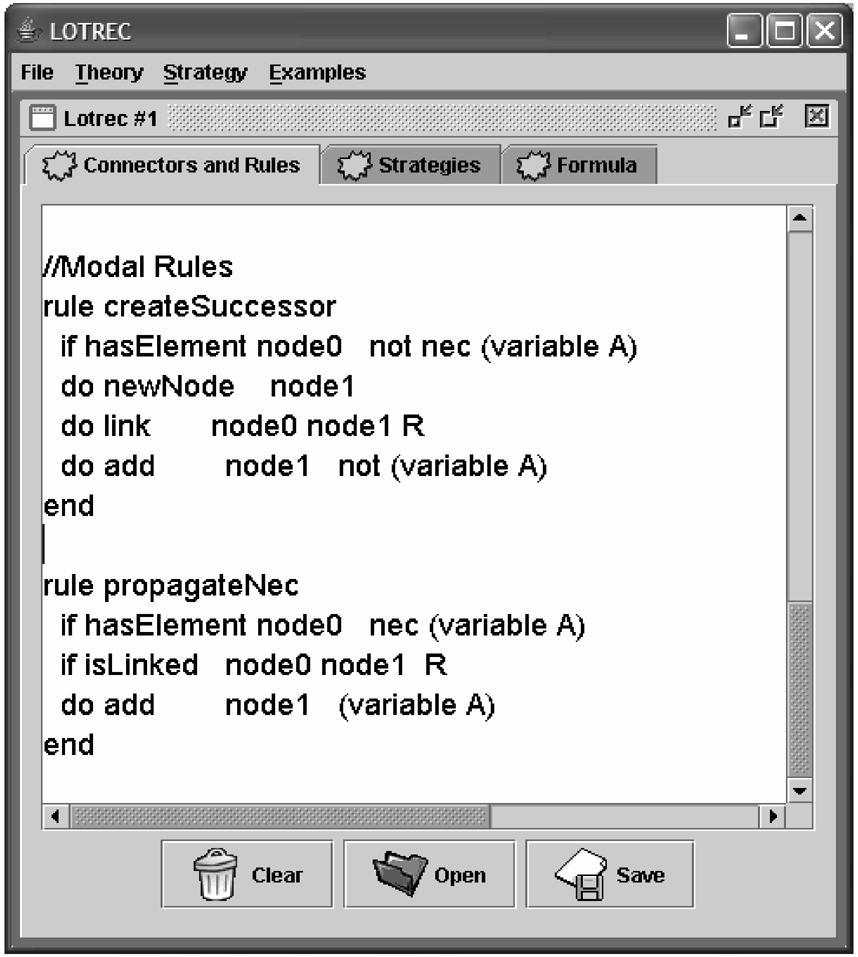


Tableau rules for K

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

- <u>KStrategy(<>P & <>Q & [](R v <>S))</u>

Modal logic KT

- accessibility relation is *reflexive*
- idea: integrate this into truth condition
  - w ||- []A iff w ||- A and forall u: Rwu implies u ||- A

Tableaux for modal logic KT

[connectors as for K…]

[rules as for K…]

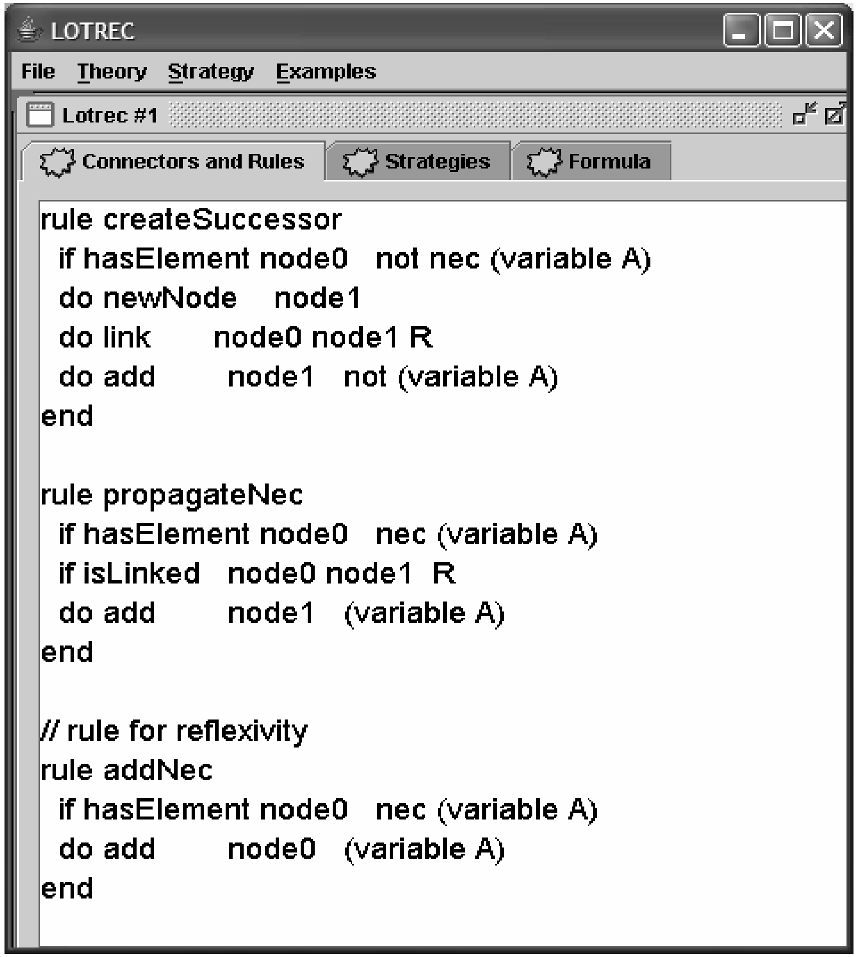
Tableaux
for modal logic KT

[connectors as for K…]

[rules as for K…]

plus: “when []A is in a node then add A to it”

- <u>KTStrategy(P & [][]~P)</u>



Tableaux for modal logic S5

accessibility relation is equivalence relation

can be supposed to be a single equivalence class

optimized tableau rules

…

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

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 []A is in a node
  then add []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 []A is in a node
  then add []A to all children"

problem: find a terminating strategy

Tableau rules for S4

- Example: w ||- []~[]P
  - add w ||- ~[]P (by rule for reflexivity)

Tableau rules for S4

- Example: w ||- []~[]P
  - add w ||- ~[]P (by rule for reflexivity)
  - create u, add Rwu, add u ||- ~P (by createSuccessor)

Tableau rules for S4

- Example: w ||- []~[]P
  - add w ||- ~[]P (by rule for reflexivity)
  - create u, add Rwu, add u ||- ~P (by createSuccessor)
  - add u ||- []~[]P (by rule for transitivity)

Tableau rules for S4

- Example: w ||- []~[]P
  - add w ||- ~[]P (by rule for reflexivity)
  - create u, add Rwu, add u ||- ~P (by createSuccessor)
  - add u ||- []~[]P (by rule for transitivity)
  - add u ||- ~[]P (by rule for reflexivity)

Tableau rules for S4

- Example: w ||- []~[]P
  - add w ||- ~[]P (by rule for reflexivity)
  - create u, add Rwu, add u ||- ~P (by createSuccessor)
  - add u ||- []~[]P (by rule for transitivity)
  - add u ||- ~[]P (by rule for reflexivity)
  - create u'
  - …

Tableau rules for S4

- Example: w ||- []~[]P
  - add w ||- ~[]P (by rule for reflexivity)
  - create u, add Rwu, add u ||- ~P (by createSuccessor)
  - add u ||- []~[]P (by rule for transitivity)
  - add u ||- ~[]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.

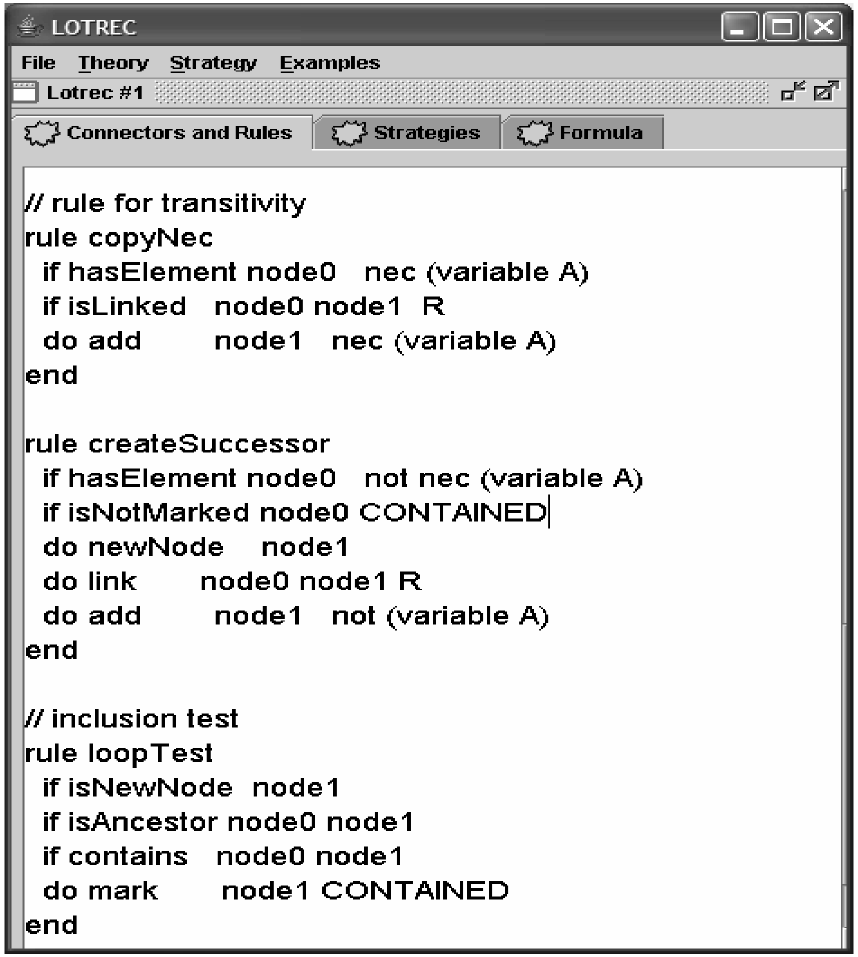


Tableau rules for S4 (ctd.)

principle:

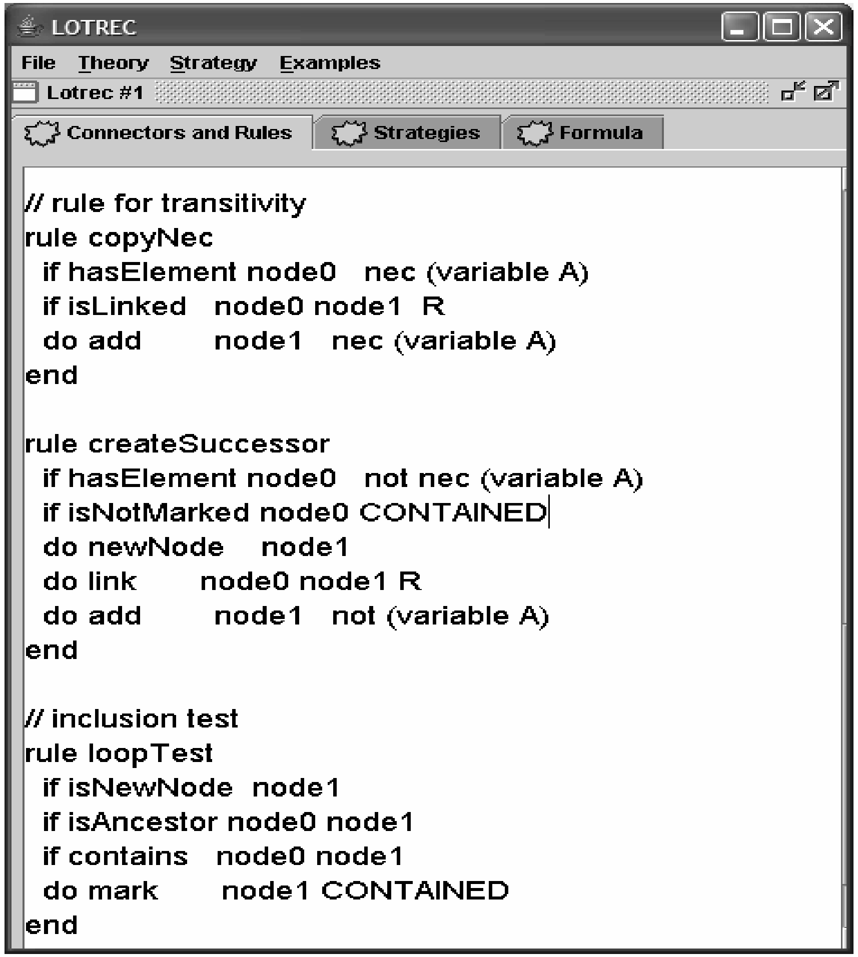
- if   a node is *included* in an ancestor

  then mark it.

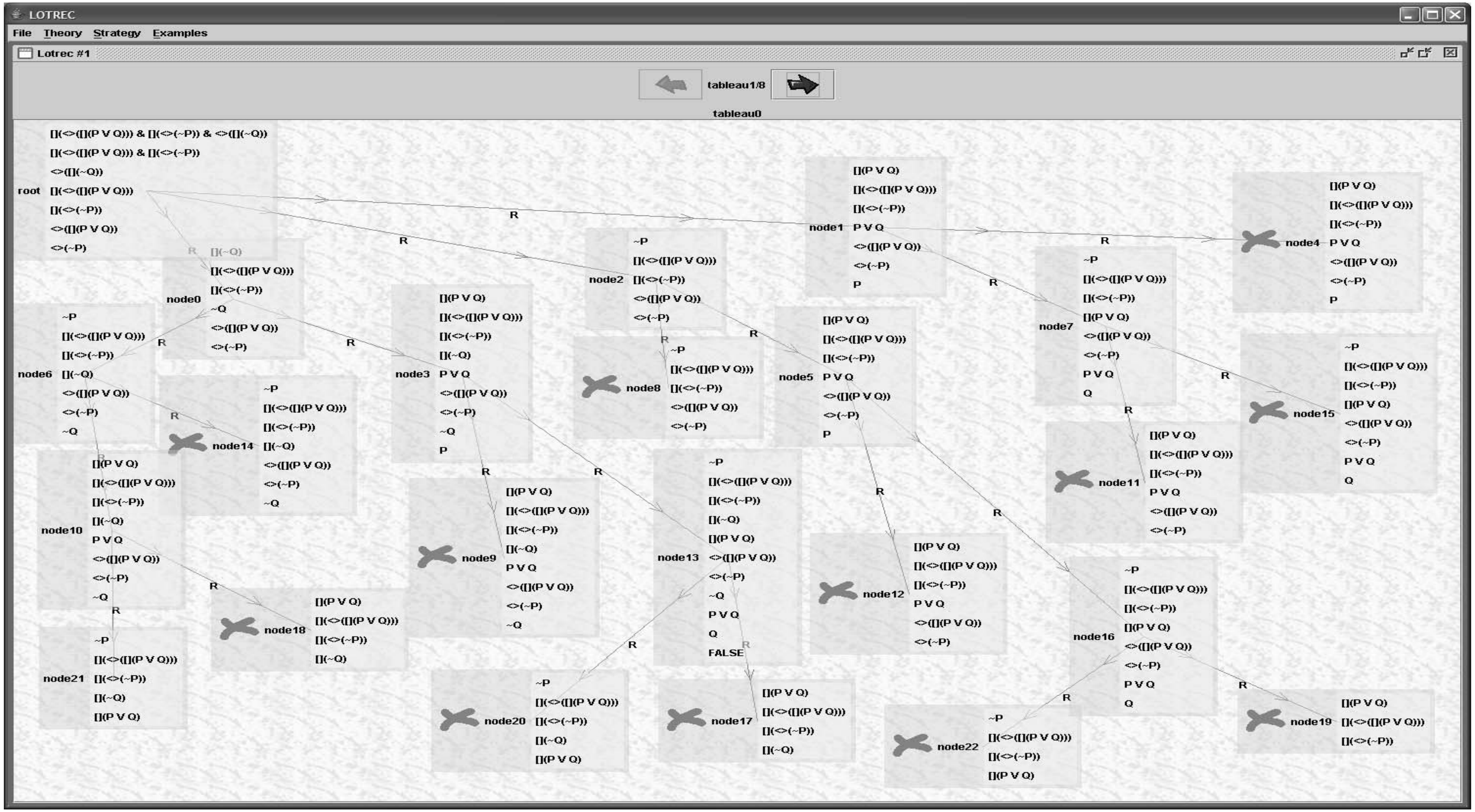
- if   a node is marked

  then block the createSuccessor rule

- <u>S4Strategy([]~[]P)</u>



S4Strategy
([]<>[] (P v Q) & []<>~P & <>[]~Q)



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

Intuitionistic logic

- no modal operators, but different semantics for implication and negation
- aim: invalidate

(~P=>FALSE) => P	*ex falso quodlibet*
P v ~P	*tertio non datur*
(~Q => ~P) => (P => Q)	*contraposition*

- R is reflexive, transitive and *hereditary*:

  if Rwu and V_w(P) = 1 then V_u(P) = 1

- similar to S4
- truth condition

  w ||- A=>B iff forall u: Rwu implies u ||- A$\rightarrow$B

Tableaux rules for intuitionistic logic

- follow translation from LJ to S4:

P’	= []P	(inheritance)
(A=>B)’	= []$(A' \rightarrow B')$	
(~A)’	= []~(A’)	

- tableaux similar to S4
- signed formulas

T(P) “P is true”

F(P) “P is false”

F(P) ≠ ~P

Tableaux rules for intuitionistic logic

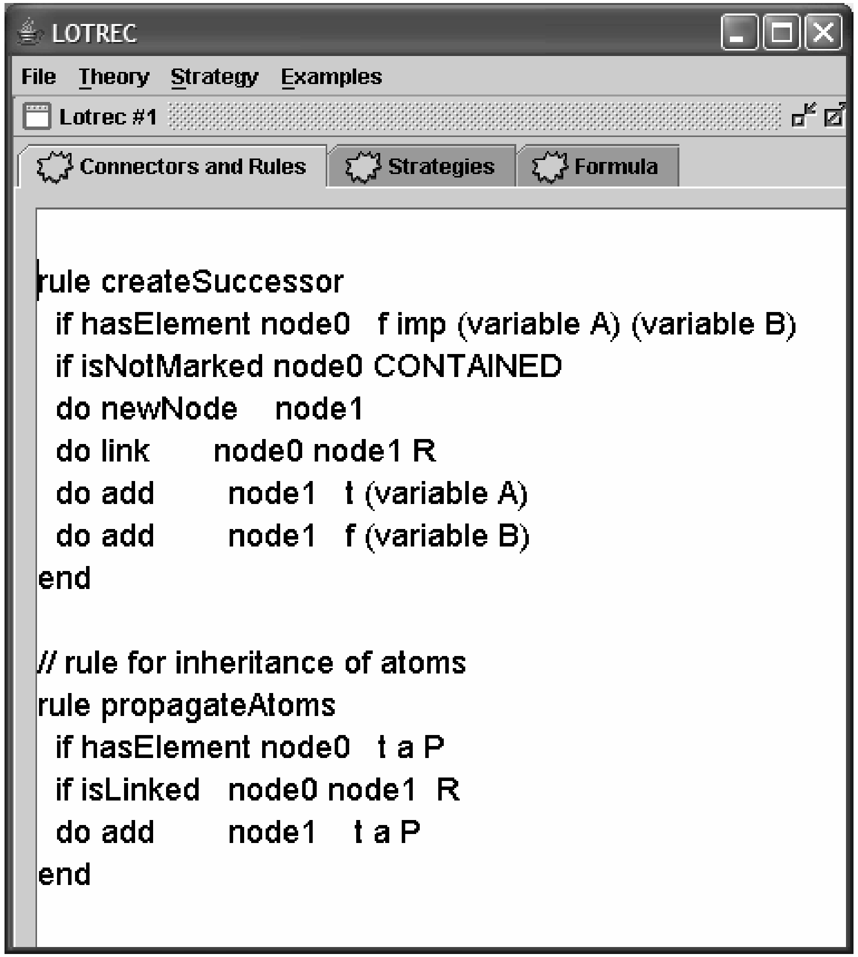
- create successor

  make A=>B false in w:

  create u, add link Rwu,

  make A false in u,

  make B true in u



Tableaux rules for intuitionistic logic

- create successor

  make A=>B false in w:

  create u, add link Rwu,

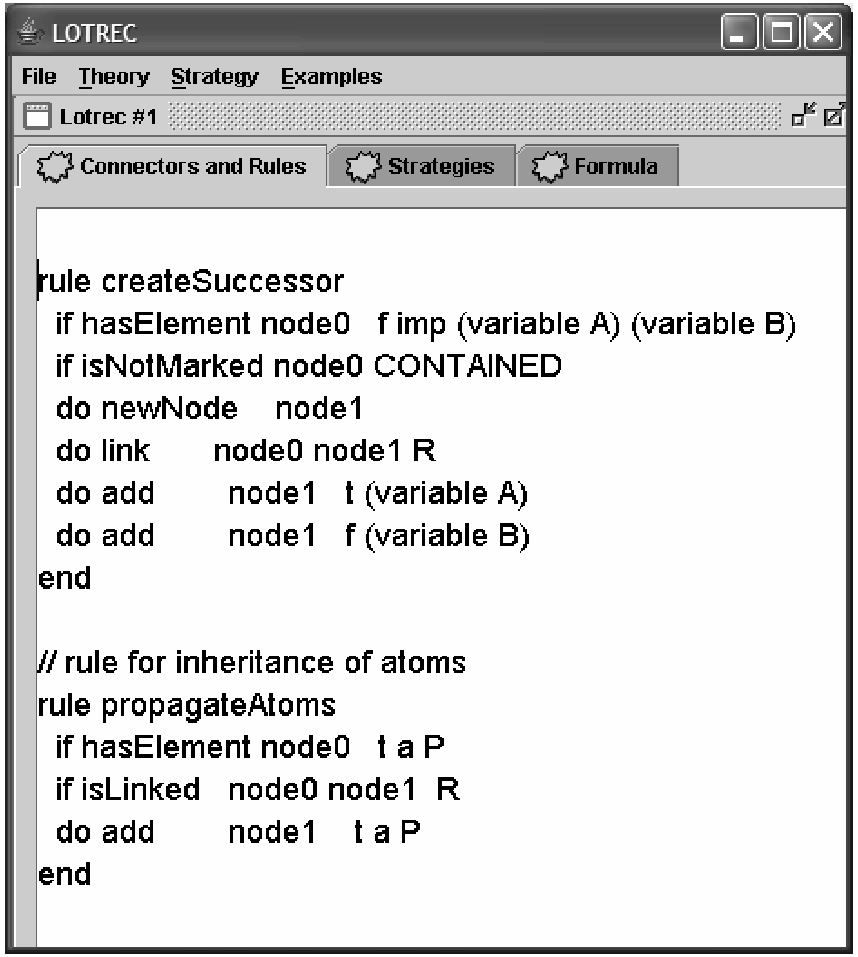
  make A false in u,

  make B true in u

- inheritance

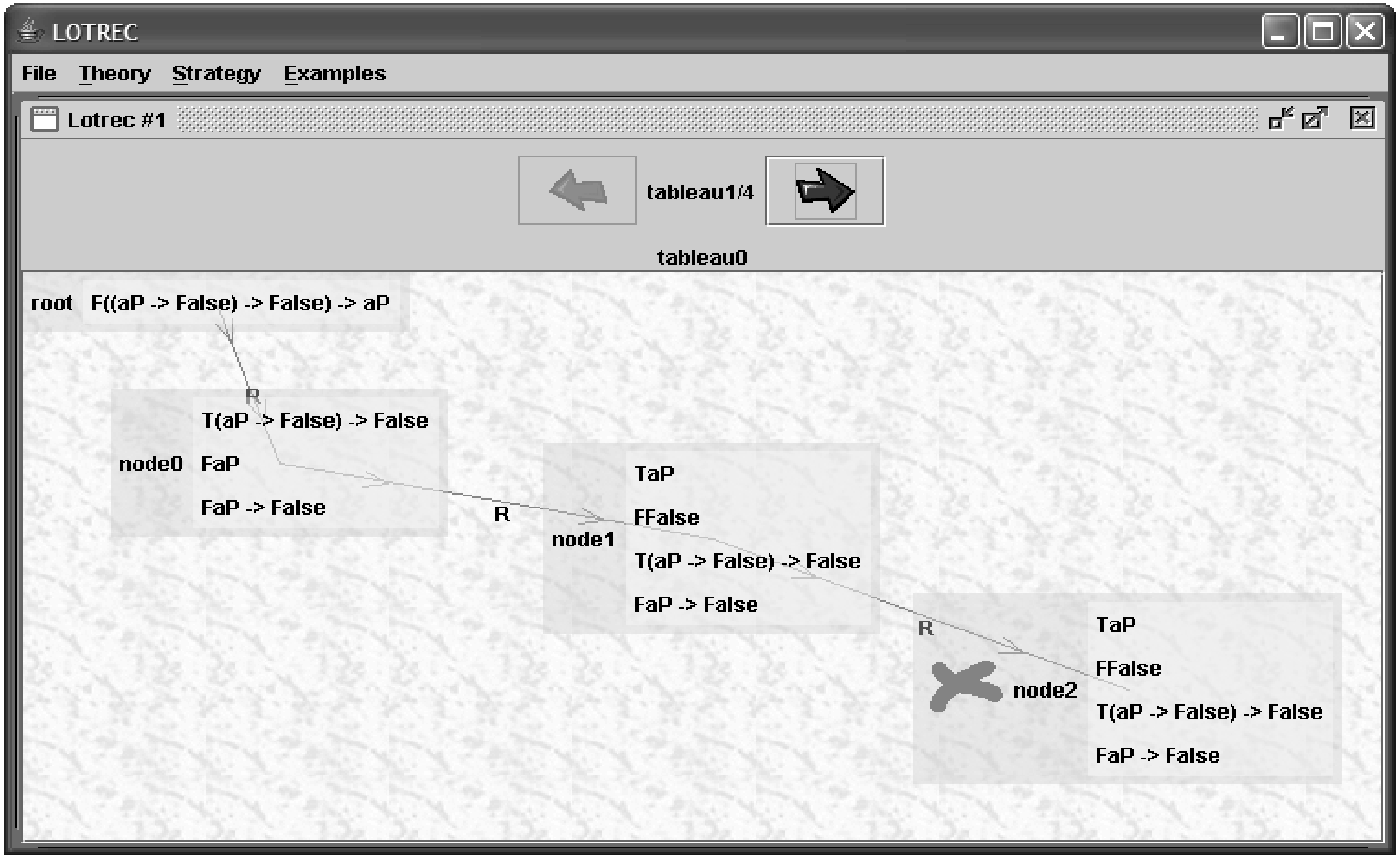
  if w ||- P and Rwu →

  then add u ||- P



Tableaux rules for intuitionistic logic: ~~P=>P

LJStrategy(((P=>False)=>False)=>P) ➔ 4 tableaux, 1 open



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

Relevant logics

- …

Paraconsistent 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

Linear Temporal Logic

- two modal operators:

  [] = always

  X = next

- R(X) is serial and deterministic
- R([]) = R(X))*

  R([]) linear order

- mix axioms:

  $[]A \leftrightarrow A \wedge X[]A$

  $<>A \leftrightarrow A \vee X<>A$

- induction axiom:

  $A \wedge [](A \rightarrow XA) \rightarrow []A$

- 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 []A to A $\wedge$ X[]A

  rewrite <>A to A v X<>A
- only create successors for X, never for <>
- 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 ||- []P

  add w ||- P∧X[]P (by mix axioms)

  add w ||- P, w ||- X[]P

  create u, add $R_X$wu, add u ||- []P

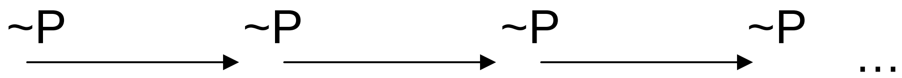
  (by propagation rule for X)

  add u ||- P∧X[]P (by mix axioms)

  add u ||- P, u ||- X[]P

  w contains u: mark u “contained”

Tableau rules for Linear Temporal Logic (ctd.)



- may result in ‘nonstandard’ models of <>P
  - ➔ “P never fulfilled”

➔ check if all <> are fulfilled!

Tableau rules for Linear Temporal Logic: example

- Example: <u>LTLStrategy(<>P)</u>

w ||- <>P

.

.

Tableau rules for Linear Temporal Logic

- Example: LTLStrategy(<>P)

w ||- <>P

w ||- P v X<>P (by mix)

Tableau rules for Linear Temporal Logic

- Example: LTLStrategy(<>P)

w ||- <>P

w ||- P v X<>P (by mix)

w ||- <>P, w ||- P

(nothing applies)

w’ ||- <>P, w’ ||- X<>P

.

Tableau rules for Linear Temporal Logic

- Example: <u>LTLStrategy(<>P)</u>

w ||- <>P

w ||- P v X<>P (by mix)

w ||- <>P, w ||- P

(nothing applies)

w' ||- <>P, w' ||- X<>P

R_Xw'u', u' ||- <>P

.

Tableau rules for Linear Temporal Logic

- Example: <u>LTLStrategy(<>P)</u>

w ||- <>P

w ||- P v X<>P (by mix)

w ||- <>P, w ||- P

(nothing applies)

w' ||- <>P, w' ||- X<>P

R_Xw'u', u' ||- <>P

u' ||- P v X<>P (by mix)

Tableau rules for Linear Temporal Logic

- Example: LTLStrategy(<>P)

w ||- <>P

w ||- P v X<>P (by mix)

w ||- <>P, w ||- P

(nothing applies)

w’ ||- <>P, w’ ||- X<>P

R_Xw’u’, u’ ||- <>P

u’ ||- P v X<>P (by mix)

u’ ||- P

u’’ ||- X<>P

Tableau rules for Linear Temporal Logic

- Example: LTLStrategy(<>P)

w ||- <>P

w ||- P v X<>P (by mix)

w ||- <>P, w ||- P

(nothing applies)

w' ||- <>P, w' ||- X<>P

R_Xw'u', u' ||- <>P

u' ||- P v X<>P (by mix)

u' ||- P

(nothing applies)

u'' ||- X<>P

contained in w'

Tableau rules for Linear Temporal Logic

- Example: <u>LTLStrategy(<>P)</u>

w ||- <>P

w ||- P v X<>P (by mix)

↙ ↘

w ||- <>P, w ||- P

(nothing applies)

w' ||- <>P, w' ||- X<>P

R_Xw'u', u' ||- <>P

u' ||- P v X<>P (by mix)

↙ ↘

u' ||- P

(nothing applies)

u'' ||- X<>P

u'' contained in w'

<>P not fulfilled

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 ∩ Female
  Father U Mother
  ~Parent
  ∃hasChild.Female "individuals having a female child"
  ∀hasChild.Female "…"
  >1 hasChild.T "individuals having more than 1 child"

- set of concepts ➔ "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 ∩ Female

  – are expanded away ➔ TBox = ∅

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

∃hasChild.Female = <hasChild>Female

∀hasChild.Female = [hasChild.Female]

Parent ∩ Female = Parent ∧ Female

Father U Mother = Father v Mother

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

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

…modal logics with number restrictions
[Fattorosi&Barnaba, van der Hoek]

Description logics

- description logic ALC:

  ~C

  $C_1 \cap C_2$

  $C_1 \cup C_2$

  ∃R.C

  ∀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

  Knw_i A   "agent i knows that A"

  [a] A   "after execution of action a, A holds"

- "product logics":

  $R_{Knwi} \circ R_a = R_a \circ R_{Knwi}$ (permutation)

  if $wR_{Knwi}u$ and wR_av then exists t such that uR_at and $vR_{Knwi}t$ (confluence)

- axiomatically:

  $Knw_i[a]A \leftrightarrow [a]Knw_iA$

  $\langle a\rangle Knw_iA \rightarrow Knw_i\langle a\rangle A$

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 ||- []A iff exists U in N(w) forall u in U: u ||- A

  non-normal modal logic EM

- can be represented by a set of relations

  w ||- []A iff exists R_i forall u ($R_i$wu implies u ||- A)

- logic EM: “non-normal”

  not valid: $[]P \wedge []Q \rightarrow [](P \wedge Q)$

  but valid: $[](P \wedge Q) \rightarrow []P \wedge []Q$

Tableau rules for EM

- …

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

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 term

  no further instantiation of (3) is possible

- decision procedure for formulas without positive $\forall \ldots \exists$

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
- 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 []A, <>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.)

- KSAT([](P&Q) & <>~P)
  - call SAT with set of clauses {[](P&Q), <>~P}
  - SAT returns:

    V([](P&Q)) = 1

    V(<>~P) = 1
  - apply createOneSuccessor and propagateNec:

    w ||- [](P&Q), w ||- <>~P, Rwu, u ||- ~P, u ||- P&Q
  - call SAT with set of clauses {P,Q,~P}
  - SAT returns:

    set of clauses unsatisfiable
  - [](P&Q) & <>~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

Thanks to…

- Mohamad Sahade
- Olivier Gasquet
- Luis Fariñas del Cerro
- Dominique Longin
- Tiago Santos de Lima
- Fabrice Evrard
- Carole Adam
- Nicolas Troquard
- Benoit Gaudou
- Ivan Varzinczak
- Bilal Saïd
- Dominique Ziegelmayer
- … and the other members of the LILaC group