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 $[]$
  – formulas $A$, $B$, …

• $M = (W,R,V)$, 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”)
  - $w \models A \land B$ \Rightarrow add $w \models A$, and add $w \models B$
  - $w \models A \lor B$ \Rightarrow add either $w \models A$, or add $w \models B$ (nondet.)
  - $w \models \neg A$ \Rightarrow “don’t add $w \models A$”??
    - $w \models \neg\neg A$ \Rightarrow add $w \models A$
    - $w \models \neg(A \lor B)$ \Rightarrow add $w \models \neg A$, and add $w \models \neg B$
    - $w \models \neg(A \land B)$ \Rightarrow add either $w \models \neg A$, or add $w \models \neg B$

- apply while possible (“downwards saturation”)
- is this a model?
  - NO if both $w \models P$ and $w \models \neg P$ (“tableau is closed”)
  - ELSE: for every $w$, if $w \models P$ put $V_w(P) = 1$, else put $V_w(P) = 0$
The basic idea: example for classical logic

\[ A = P \land \neg(P \land Q) \]

- applying truth conditions:
  1. \( w \models P \land \neg(P \land Q) \)
  2. \( w \models P \land \neg(P \land Q), w \models P, w \models \neg(P \land Q) \)
  3. \( w \models P \land \neg(P \land Q), w \models P, w \models \neg(P \land Q), w \models \neg P \) (nondet.)

- no more truth condition applies
- can’t be a model:
  both \( w \models P \) and \( w \models \neg P \)
- backtrack on nondeterministic choices
The basic idea: example for classical logic (ctd.)

- 1st downward saturated graph for
  \( A = P \land \neg(P \land Q) \)
  \[ \Rightarrow \] not a model
  (contains \( P \) and \( \neg P \))
The basic idea: example for classical logic (ctd.)

- 1st downward saturated set for
  \[ A = P \land \neg (P \land Q) \]
  \[ \Rightarrow \text{not a model} \]
  (contains \( P \) and \( \neg P \! \))

- 2nd downward saturated set for
  \[ A = P \land \neg (P \land Q) \]
  \[ \Rightarrow \text{is a model of} \ A \]
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
  \( w \models []A \) ➞ for all \( u \) such that \( Rwu \), add \( u \models A \)
  \( w \models <>A \) ➞ add some new \( u \), add \( Rwu \), add \( u \models A \)
  \( w \models ~[]A \) ➞ add some new \( u \), add \( Rwu \), add \( u \models ~A \)
  \( w \models ~<>A \) ➞ ...

• “downwards saturated graph”: is this a model?
The basic idea:
example for modal logic

\[ A = P \land \neg[]P \]

• applying tableau rules:
  1. \( w \vdash P \land \neg[]P \)
  2. \( w \vdash P \land \neg[]P, w \vdash \neg A \)
  3. \( w \vdash P \land \neg[]P, w \vdash \neg A \), \( w \vdash \neg[]P \), Ruw, u \vdash \neg P \\
  no more tableau rule applies
  \( \Rightarrow \) never both \( w \vdash A \) and \( w \vdash \neg A \) (“open tableau”)

• model can be built: \( M = (W,R,V) \)
  set of worlds \( W \):
  \( W = \{w,u\} \)
  accessibility relation \( R \):
  \( R[w,u] \)
  valuation \( V \):
  \( V_w(P) = 1, V_u(P) = 0 \)
The basic idea: example for modal logic (ctd.)

- premodel for
  \[ A = P \land \neg[]P \]
  \[ \Rightarrow \text{not closed} \]
  \[ \Rightarrow \text{is a model of } A \]
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]:

\[(P_1 \lor P_2 \lor P_3) \land (P_1 \lor P_2 \lor \neg P_3) \land (P_1 \lor \neg P_2 \lor P_3) \land \ldots\]

there are formulas with \(n\) atoms of length \(O(2^n)\)

\(\Rightarrow\) truth tables have \(2^n\) rows

\(\Rightarrow\) 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
  ...
  ```

- 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

```plaintext
// 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 Stop:
if there is an explicit contradiction then stop exploring the tableau

```
// 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 $\neg\neg A$ 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:

```plaintext
firstrule rule_1 \ldots rule_n end
   "apply first applicable rule and stop"
allrules rule_1 \ldots 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

⇒ “fair strategy”
Strategy for classical logic: example

\[ \text{CPLStrategy}(P \& \neg(P \& Q)) \]
Strategy for classical logic: example (ctd.)

\[ \text{CPLStrategy}(P \& \neg(P \& Q)) = \{ T1, T2 \} \]
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 \( \neg \neg A \) in node then A in node  
    if A&B in node then A in node and B in node  
    if \( \neg (A&B) \) in node then \( \neg A \) in node or \( \neg B \) in node  
    (because allRules strategy is fair: every rule eventually applies)
  – Build a CPL model from G: 
    \[ V_{\text{node}}(P) = 1 \text{ iff } P \text{ appears in node} \]
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:
    \( V_{\text{node}}(P) = 1 \) 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:
  \[ w \models []A \iff \text{forall } u: Rwu \implies u \models A \]
  \[ w \models <>A \iff \text{exists } u: Rwu \text{ and } u \models A \]
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
connectors: not, and, nec

[some rules for classical logic…]

propagateNec:
if nec A is in node0
node0 is linked node1 R
then add node1 A
end
Tableaux for K

• … plus rules for the definable connectives

• \texttt{KStrategy(<>P \& <>Q \& [](R \lor <>S))}
Modal logic KT

• accessibility relation is reflexive
• idea: integrate this into truth condition
  – $w \models []A$ iff $w \models A$ and forall $u$: $Rwu$ implies $u \models A$
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”

• **KTStrategy(P & [][]~P)**
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 []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: \( w \vdash \Box \neg \Box P \)
  - add \( w \vdash \neg \Box P \) (by rule for reflexivity)
Tableau rules for S4

• Example: $w \vdash []\neg[]P$
  - add $w \vdash \neg[]P$ (by rule for reflexivity)
  - create $u$, add $Ru$, add $u \vdash \neg P$ (by createSuccessor)
Tableau rules for S4

- Example: $w \models [] \neg [] P$
  - add $w \models \neg [] P$ (by rule for reflexivity)
  - create $u$, add $Rwu$, add $u \models \neg P$ (by createSuccessor)
  - add $u \models [] \neg [] P$ (by rule for transitivity)
Tableau rules for S4

• Example: $w \vdash \Box \neg \Box P$
  – add $w \vdash \neg \Box P$ (by rule for reflexivity)
  – create $u$, add $Rwu$, add $u \vdash \neg P$ (by createSuccessor)
  – add $u \vdash \Box \neg \Box P$ (by rule for transitivity)
  – add $u \vdash \neg \Box P$ (by rule for reflexivity)
Tableau rules for S4

• Example: w \vdash []\neg[]P
  – add w \vdash \neg[]P \quad \text{(by rule for reflexivity)}
  – create u, add Rwu, add u \vdash \neg P \quad \text{(by createSuccessor)}
  – add u \vdash []\neg[]P \quad \text{(by rule for transitivity)}
  – add u \vdash \neg[]P \quad \text{(by rule for reflexivity)}
  – create u’
  – …
Tableau rules for S4

• Example: \( w \models []\neg[]P \)
  – add \( w \models \neg[]P \) (by rule for reflexivity)
  – create \( u \), add \( Rwu \), add \( u \models \neg P \) (by createSuccessor)
  – add \( u \models []\neg[]P \) (by rule for transitivity)
  – add \( u \models \neg[]P \) (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.
principle:
• if a node is *included* in an ancestor then mark it.
• if a node is marked then block the createSuccessor rule

• **S4Strategy([]~[P])**
S4Strategy

$$([-][-] (P \lor Q) \land [\mathbf{<>}][-P]) \land [\mathbf{<>}][-Q])$$
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:
  
  $[] = \text{always}$
  
  $X = \text{next}$

• $R_X$ is serial and deterministic

• $R[] = R_X^*$
  
  $R(\[])$ linear order

• fixpoint axioms:
  
  $\[]A \iff A \land X\[]A$
  
  $<>A \iff A \lor X<>A$

• least fixpoint axiom:
  
  $A \land [](A \rightarrow XA) \rightarrow []A$

• 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 $\mathcal{[]A}$ to $A \land X\mathcal{[]A}$
  - rewrite $\mathcal{<>A}$ to $A \lor X\mathcal{<>A}$

• 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: $w \models []P$

  add $w \models P \land X[]P$  (by fixpoint axioms)
  add $w \models P$, $w \models X[]P$

  create $u$, add $R_xwu$, add $u \models []P$

  (by propagation rule for $X$)

  add $u \models P \land X[]P$  (by mix axioms)
  add $u \models P$, $u \models X[]P$

$w$ contains $u$: mark $u$ “contained”
Tableau rules for Linear Temporal Logic (ctd.)

\[ \neg P \rightarrow \neg P \rightarrow \neg P \rightarrow \neg P \rightarrow \ldots \]

- 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(<>P)`
  
  \[ w \models <>P \]
Tableau rules for Linear Temporal Logic

• Example: \texttt{LTLStrategy(<>P)}

\begin{align*}
w & 
\vDash \ \text{<>P} \\
w & 
\vDash \ P \vee X\text{<>P} \quad \text{(by fix)}
\end{align*}
Tableau rules for Linear Temporal Logic

- **Example:** \texttt{LTLStrategy(<>P)}

\[
\begin{align*}
\text{w} \models & \text{ <>P} \\
\text{w} \models & \text{ P v X<>P} \\
\text{w} \models & \text{ <>P, w} \models \text{ P} \\
\text{w'} \models & \text{<>P, w'} \models \text{ X<>P} \\
\end{align*}
\]

(by fix)

\textit{(nothing applies)}
Tableau rules for Linear Temporal Logic

• Example: $\text{LTLStrategy}(<>P)$

$w \models <>P$

$w \models P \lor X<>P$ (by fix)

$w \models <>P, w \models P$

$w' \models <>P, w' \models X<>P$

$(nothing applies)$

$R_X w'u', u' \models <>P$

.
Tableau rules for Linear Temporal Logic

- **Example:** \texttt{LTLStrategy(<>P)}

\[
\begin{align*}
& w \models \textit{<>}P \\
& w \models P \lor X<>P \quad \text{(by fix)}
\end{align*}
\]

\[
\begin{align*}
& (\text{nothing applies}) \\
& R_x w' u', u' \models \textit{<>}P \\
& u' \models P \lor X<>P \quad \text{(by fix)}
\end{align*}
\]
Tableau rules for Linear Temporal Logic

• Example: \texttt{LTLStrategy(<>P)}

\[ w \models <>P \]
\[ w \models P \lor X<>P \quad \text{(by fix)} \]
\[ w \models <>P, \quad w \models P \quad w' \models <>P, \quad w' \models X<>P \]
\[ (\text{nothing applies}) \]
\[ R_xw'u', \quad u' \models <>P \]
\[ u' \models P \lor X<>P \quad \text{(by fix)} \]
\[ u' \models P \quad u'' \models X<>P \]
Tableau rules for Linear Temporal Logic

- Example: \( \text{LTLStrategy}(<>P) \)

\[
\begin{align*}
\text{w} & \models <>P \\
\text{w} & \models P \lor X<>P \\
\text{w} & \models P, \text{w} \models P, \text{w'} & \models <>P, \text{w'} & \models X<>P \\
\text{w} & \models <>P, \text{w} \models P & \text{w'} & \models <>P, \text{w'} & \models X<>P \\
\text{w} & \models <>P, \text{w} \models P & \text{w'} & \models X<>P \\
\text{w} & \models P, \text{w} \models P, \text{w'} & \models <>P, \text{w'} & \models X<>P \\
\end{align*}
\]

(by fix)
Tableau rules for Linear Temporal Logic

- **Example:** \texttt{LTLStrategy(<>P)}

\[
\begin{align*}
\text{w} & \models <>\text{P} \\
\text{w} & \models \text{P} \lor X<>\text{P} \quad \text{(by fix)} \\
\text{w} & \models <>\text{P}, \text{w} \models \text{P} \quad \text{w}' \models <>\text{P}, \text{w}' \models X<>\text{P} \\
& \quad \text{(nothing applies)} \quad \text{R}_X \text{w}'u', \text{u}' \models <>\text{P} \\
& \quad \text{u}' \models \text{P} \quad \text{u}' \models \text{P} \lor X<>\text{P} \quad \text{(by fix)} \\
& \quad \text{u'} \models <>\text{P} \quad \text{u''} \models X<>\text{P} \\
& \quad \text{(nothing applies)} \quad \text{u''} \text{ contained in w'} \\
& \quad \text{<>P not fulfilled} \\
\end{align*}
\]
Propositional dynamic logic (PDL)

- **two kinds of expressions**
  - formulas:
    \[ A ::= P \mid \neg A \mid A \land B \mid \lbrack \pi \rbrack A \]
  - programs:
    \[ \pi ::= a \mid \pi_1;\pi_2 \mid \pi_1 \cup \pi_2 \mid \pi^* \mid A? \]

- **in the models:** \( R \) interprets programs
  \[
  \begin{align*}
  R(\pi_1;\pi_2) &= R(\pi_1);R(\pi_2) \\
  R(\pi_1 \cup \pi_2) &= R(\pi_1) \cup R(\pi_2) \\
  R(\pi^*) &= (R(\pi))^* \\
  R(A?) &= \{<w,w> : w \models A\}
  \end{align*}
  \]
Tableaux for PDL

• **similar to LTL:**
  – expand \([\pi^*]A\) to \(A \land [\pi][\pi^*]A\)
  – 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:
    \[
    \begin{align*}
    [\pi_1;\pi_2]A & \quad \iff \quad [\pi_1][\pi_2]A \\
    [\pi_1 \cup \pi_2]A & \quad \iff \quad [\pi_1]A \land [\pi_2]A \\
    [A?]B & \quad \iff \quad A \rightarrow B
    \end{align*}
    \]
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: $C_1 \cap \neg C_2$ 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 $C_1 \cap C_2$

$\Rightarrow$ all reasoning tasks reduce to concept satisfiability
Description logics

• translation of concepts into modal logics

\[ \exists \text{hasChild}.\text{Female} = \langle \text{hasChild}\rangle\text{Female} \]
\[ \forall \text{hasChild}.\text{Female} = [\text{hasChild}.\text{Female}] \]
\[ \text{Parent} \cap \text{Female} = \text{Parent} \land \text{Female} \]
\[ \text{Father} \cup \text{Mother} = \text{Father} \lor \text{Mother} \]
\[ <2 \text{hasChild}.T = [\text{hasChild}]_2 T \]
\[ \geq 2 \text{hasChild}.T = \langle \text{hasChild}\rangle_2 T \]

…modal logics with number restrictions

[Fattorosi&Barnaba, van der Hoek]
Description logics

• description logic ALC:
  \( \neg C \)
  \( C_1 \cap C_2 \)
  \( C_1 \cup C_2 \)
  \( \exists R.C \)
  \( \forall R.C \)
  = multimodal K

• description logic \( ALC_{reg} = ALC + \text{regular expressions on roles} \)
  = 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
  \[ \text{Knw}_i A \quad \text{“agent i knows that A”} \]
  \[ [a] A \quad \text{“after execution of action a, A holds”} \]

- “product logics”:  
  \[ R_{\text{Knw}_i} \circ R_a = R_a \circ R_{\text{Knw}_i} \]  
  (permutation)
  
  if \( w R_{\text{Knw}_i} u \) and \( w R_a v \) then exists \( t \) such that \( u R_a t \) and \( v R_{\text{Knw}_i} t \)  
  (confluence)

- axiomatically:
  \[ \text{Knw}_i [a] A \leftrightarrow [a]\text{Knw}_i A \]
  \[ \langle a \rangle \text{Knw}_i A \rightarrow \text{Knw}_i \langle a \rangle A \]

  tableaux: …

  ➔ problem: combination with transitivity
Belief-Desire-Intention logics

• [Bratman, Rao&Georgeff]
• 3 modal operators
  \[ \text{Bel}_i A \quad \text{Desire}_i A \quad \text{Intend}_i A \]
  \[
  \begin{align*}
  \text{Bel}_i A & \quad \text{“agent i believes that A”} \\
  \text{Desire}_i A & \quad \text{“agent i desires that A”} \\
  \text{Intend}_i A & \quad \text{“agent i intends that A”}
  \end{align*}
  \]
• 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: \( N : W \rightarrow 2^{2^W} \)
  
  \[ w \models \Box A \text{ iff exists } U \text{ in } N(w) \text{ forall } u \text{ in } U : u \models A \]
  
  non-normal modal logic EM

• can be represented by a set of relations
  
  \[ w \models \Box A \text{ iff exists } R_i \text{ forall } u : (R_iwu \implies u \models A) \]

• 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?
  \( \forall x \ p(x) \land \neg p(a) \) is unsatisfiable
  \( \forall x \ p(x) \land \exists x \ \neg p(x) \) is unsatisfiable

- naïve implementation [Beth, Smullyan]:
  if hasElement node0 forall x A(x)
  do createTerm t
  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
  do add node0 A(t)

• ex.: $\exists x \ p(x,x) \land \exists x \forall y \neg p(x,y)$ satisfiable
  1. $\exists x \ p(x,x)$
  2. $\exists x \forall y \neg p(x,y)$
  3. $\forall y \neg p(a,y)$ (2), new constant
  4. $\neg p(a,a)$ (3), only Herbrand term
  5. $p(b,b)$ (1), new constant
  6. $\neg p(a,b)$ (3), Herbrand term

no further instantiation of (3) is possible

• decision procedure for formulas without positive $\forall$ … $\exists$
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. …

\[ \rightarrow \text{loops} \]
Free-variable tableaux with unification

- 2nd solution: don’t instantiate at all
  - work with free variables
  - runtime skolemization of existential quantifiers
  - term unification
- ex.: \( \forall x \exists y \, p(x,y) \land \forall x \exists y \, \neg p(x,y) \) satisfiable
  1. \( \forall x \exists y \, p(x,y) \)
  2. \( \forall x \exists y \, \neg p(x,y) \)
  3. \( \exists y \, p(x_1,y) \) from (1), replace x by free \( x_1 \)
  4. \( \exists y \, \neg p(x_2,y) \) from (2), replace x by free \( x_2 \)
  5. \( p(x_1,f(x_1)) \) from (3), Skolem function \( f(x_1) \)
  6. \( \neg p(x_2,g(x_2)) \) 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) \& <>\neg P)**
  - call SAT with set of clauses {\[](P\&Q), <>\neg P}
  - SAT returns:
    - $V(\[](P\&Q)) = 1$
    - $V(<>\neg P) = 1$
  - apply createOneSuccessor and propagateNec:
    - $w \models [](P\&Q)$, $w \models <>\neg P$, $R_{wu}$, $u \models \neg P$, $u \models P\&Q$
  - call SAT with set of clauses {P, Q, \neg P}
  - SAT returns:
    - *set of clauses unsatisfiable*
  - \[](P\&Q) \& <>\neg 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