Synthesis, Refinements and Search Strategies for Semantic Tableaux with Blocking

Renate A. Schmidt
School of Computer Science
The University of Manchester
1 July 2012

Tableau-based deduction

- Has a long tradition and is a well established method in AR
- Approach can be successfully used for a large number of logics
- Many implemented systems
- Multitude of different approaches

<table>
<thead>
<tr>
<th>First-order logic</th>
<th>Smullyan ground sentence tableau, free-variable tableau, connection tableau, disconnection tableau, hypertableau, ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modal, description, hybrid, intuitionistic logics, ...</td>
<td>ground semantic tableau, tableau avoiding reference to semantics ...</td>
</tr>
</tbody>
</table>

- **Our focus** Ground semantic tableau calculi with blocking for mainly non-classical logics

The essence of tableau-based deduction

- Refutation approach, testing satisfiability (constructing a model)
- Goal-directed
- Rules break down formulae
- Rules for each logical operator
- Branching rules \( \leadsto \) derivations are trees

Process of developing a tableau prover for some logic

- Define a sound and complete calculus
  - Not difficult for semantically defined logics
  - Calculi can be synthesised
- Making tableau calculi effective
  - Refining the rules
- Ensure termination for decidable logics
  - Devise blocking technique
  - Various possibilities and challenges
- Decide how to perform search
  - Issues of turning the non-deterministic calculus it into a deterministic procedure
  - Search strategies
Modal logic S4

Propositional modal logic = propositional logic plus □

**Formulae:** φ, ψ → p_i | ⊥ | ¬φ | φ ∧ ψ | □φ

**Semantics:** Kripke model $\mathcal{M} = (W, R, ν)$

- $\mathcal{M}, x \models p_i$ if $x \in ν(p_i)$
- $\mathcal{M}, x \not\models ⊥$
- $\mathcal{M}, x \models ¬φ$ if $\mathcal{M}, x \not\models φ$
- $\mathcal{M}, x \models φ ∧ ψ$ if $\mathcal{M}, x \models φ$ and $\mathcal{M}, x \models ψ$
- $\mathcal{M}, x \models □φ$ if for all $R$-successors $y$ of $x$ $\mathcal{M}, y \models φ$

$R$ is a pre-order, i.e., reflexive and transitive

---

Step 2: Extracting tableau rules

- **Conversion for left-to-right definition of □:**
  \[
  ∀x \ [ ν(□φ, x) → ∀y \ (R(x, y) → ν(φ, y))] \\
  ν(□φ, x) → ¬R(x, y) \lor ν(φ, y) \\
  ν(□φ, x) \\
  \frac{¬R(x, y) \lor ν(φ, y)}{}
  \]

- **Conversion for right-to-left definition of □:**
  \[
  ∀x \ [ ν(□φ, x) ← ∀y \ (R(x, y) → ν(φ, y))] \\
  ∀x \ [ ν(□φ, x) → ¬∀y \ (R(x, y) → ν(φ, y))] \\
  ν(□φ, x) → R(x, f(φ, x)) ∧ ¬ν(φ, f(φ, x)) \\
  ¬ν(□φ, x) \\
  \frac{R(x, f(φ, x)) \lor ¬ν(φ, f(φ, x))}{f(φ, x) = \text{Skolem term uniquely associated with } ¬ν(□φ, x)}
  \]

---

A synthesised tableau calculus for S4

- **Decomposition rules**
  \[
  ν(⊥, x) \quad ν(¬φ, x) \quad ν(φ, x) \\
  \\
  ν(φ ∧ ψ, x) \quad ν(φ ∨ ψ, x) \quad ν(□φ, x) \\
  \\
  ν(□φ, x) → R(x, f(φ, x)) \lor ν(φ, f(φ, x))
  \]

- **Closure rules**
  \[
  ν(φ, x) \quad ν(¬φ, x) \\
  \\
  R(x, y), ¬R(x, y) \\
  \\
  R(x, f(φ, x)), ν(φ, f(φ, x))
  \]

- **Theory rules**
  \[
  R(x, x) \\
  \\
  ⊥ \quad R(x, y) \lor ¬R(y, z) \lor R(x, z)
  \]

Definition of rule application is so that all rules are grounding
Refinement 1: How to get rid of $\nu$ symbols?

- Define $\nu$ as a logical operator in the semantic specification
  \[ \forall x \left[ \nu(s, x) \equiv \nu(x_0(s)) \right] \]

  Transformation of rules:
  \[ \begin{align*}
  \nu(\phi_1 \land \phi_2, x) & \implies s : \phi_1 \land \phi_2 \\
  \nu(\phi_1, x), \nu(\phi_2, x) & \implies s : \phi_1, s : \phi_2 \\
  \neg \nu(\phi_1 \land \phi_2, x) & \implies s : \neg(\phi_1 \land \phi_2)
  \end{align*} \]

  : is not needed if the logic includes the $\boxdot$ operator of hybrid logic

- Gives transformation to labelled prefix tableau calculus

A labelled prefix tableau calculus for S4

**Decomposition rules**

\[ \begin{align*}
  s : \bot & \implies \bot \\
  s : \phi \land \psi & \implies s : \phi, s : \psi \\
  s : \neg \phi & \implies s : \neg \phi \\
  s : \boxdot \phi & \implies R(s, f(\phi, s)), f(\phi, s) : \phi \\
  \neg R(s, t) | t : \phi & \implies \neg R(s, t) \\
  R(s, t), \neg R(s, t) & \implies \bot \\
  R(s, f(\phi, s)), f(\phi, s) : \phi & \implies \bot \\
  \neg R(s, t) | - R(t, u) | R(s, u) & \implies \bot
  \end{align*} \]

**Closure rules**

\[ \begin{align*}
  s : \phi, s : \neg \phi & \implies \bot \\
  \neg R(s, t) | t : \phi & \implies \neg R(s, t) \\
  R(s, t), \neg R(s, t) & \implies \bot
  \end{align*} \]

**Theory rules**

\[ \begin{align*}
  R(s) & \implies R(s) \\
  \neg R(s) & \implies \bot \\
  \neg R(s, t) | - R(t, u) | R(s, u) & \implies \bot
  \end{align*} \]

Standard rules for S4

- Standard ML tableau calculi include
  \[ \begin{align*}
  s : \boxdot \phi, R(s, t) & \implies \bot \\
  t : \phi & \implies \neg R(s, t) \mid t : \phi \\
  R(s, t), R(t, u) & \implies R(s, u)
  \end{align*} \]

  Less branching reduces the search space

  - These examples suggest a general refinement principle

Rule refinement

Suppose $\text{Tab}$ includes this rule, where $X_1 = \{F_1, \ldots, F_k\}$

\[ \rho = \frac{X_0}{X_1 \mid \cdots \mid X_m} \]

- Refinement $\text{Tab'}$ of $\text{Tab} = \text{Tab}$ with $\rho$ replaced by $\{\rho_1, \ldots, \rho_k\}$

  \[ \rho_j = \frac{X_0 \cup \{\neg F_j\}}{X_2 \mid \cdots \mid X_m} \quad (j = 1, \ldots, k) \]

  $\sim$ denotes complement

- Some properties
  - Each $\rho_j$ is sound, if $\rho$ is sound
  - Each $\rho_j$ is derivable in $\text{Tab}$
  - In general, $\rho$ is not derivable in $\text{Tab'}$
### Soundness and completeness of refined tableau calculi

**Dagger condition** – sufficient condition for completeness of Tab':

In any Tab'-tableau derivation and every open branch B, if $E_1, \ldots, E_k$ belong to B and $X_0 \alpha = \{E_1, \ldots, E_k\}$ and each $E_i$ holds in $I(B)$, then

$$I(B) \models X_i \sigma \quad \text{for some } i = 1, \ldots, m$$

**Theorem (Refinement)**

1. Tab' is sound whenever Tab is sound.
2. If Tab is complete and the dagger condition holds in any Tab'-tableau derivation then Tab' is complete.

### Refining rules for S4

**Dagger condition is true for**

\[
\begin{align*}
\text{s : } \square \phi & \quad \Rightarrow \quad \text{s : } \square \phi, \ R(s, t) \\
\neg R(s, t) & \quad \Rightarrow \quad \text{t : } \phi \\
\neg R(s, t) & \quad \Rightarrow \quad \text{R(s, t), R(t, u)} \\
\end{align*}
\]

**Dagger condition is not true for**

\[
\begin{align*}
\text{s : } \neg(\phi \land \psi) & \quad \Rightarrow \quad \text{s : } \neg(\phi \land \psi), \ s : \phi \\
\text{s : } \neg\phi & \quad \Rightarrow \quad \text{s : } \neg\psi \\
\end{align*}
\]

But it is true for

\[
\begin{align*}
\text{s : } \neg(p \land \psi) & \quad \Rightarrow \quad \text{s : } \neg(p \land \psi), \ s : p \\
\text{s : } \neg p & \quad \Rightarrow \quad \text{s : } \neg\psi \\
\end{align*}
\]

### Synthesised calculus for $K(m)(\neg)$

**Decomposition rules**:

\[
\begin{align*}
\text{s : } \neg\phi & \quad \Rightarrow \quad \text{s : } \neg\phi, \ \text{s : } \phi \\
\text{s : } \phi \land \psi & \quad \Rightarrow \quad \text{s : } \neg(\phi \land \psi), \ s : \phi \\
\text{s : } \neg\phi & \quad \Rightarrow \quad \text{s : } \neg\psi \\
\text{s : } \phi & \quad \Rightarrow \quad \text{R(s, t), R(t, u)} \\
\end{align*}
\]

**Closure rules**:

\[
\begin{align*}
\text{s : } \neg\phi & \quad \Rightarrow \quad \text{s : } \neg\phi, \ \text{R(s, t), } \neg R(s, \tau) \\
\text{s : } \phi & \quad \Rightarrow \quad \text{R(s, s)} \\
\end{align*}
\]

**Theory rules**:

\[
\begin{align*}
\text{R(s, s)} & \quad \Rightarrow \quad \text{R(s, t), R(t, u)} \\
\end{align*}
\]

Only positive R-literals derived $\Rightarrow$ S4 has a kind of tree model property

### Question:

Can the [-] rule be replaced by the refined non-branching version?
## Rules synthesised from an alternative spec. for $K_m(\neg)$

### Decomposition rules:

$$
\begin{align*}
\text{Decomposition rules:} & \\
\text{s: } & \neg\neg\phi & \text{s: } & \phi & \text{s: } & \phi \lor \psi & \text{s: } & \neg(\phi \lor \psi) \\
\text{s: } & \neg\phi, \text{ s: } & \neg\psi \\
(s, t) : & \neg\alpha & t : & \phi & (s, t) : & \neg\alpha, (s, t) : & \neg\alpha \\
(t, s) : & \alpha & t : & \phi
\end{align*}
$$

### Closure rules:

$$
\begin{align*}
\text{Closure rules:} & \\
\text{s: } & \phi, \text{ s: } & \neg\psi \\
\text{(s, t): } & \alpha, (s, t): & \neg\alpha
\end{align*}
$$

Now, the dagger condition holds for the $[\cdot]$ rule.

## Termination via blocking

### General idea of blocking

Use the tableau procedure to find finite models through reusing terms or identifying terms

### Reusing terms

- Standard loop-checking mechanisms
- Subset or equality blocking
- Ancestor or anywhere blocking
- Static or dynamic blocking

### Many other techniques

- Pairwise blocking
- Core blocking
- Pattern-based blocking
- $\delta^*$-rule

### Identifying terms and equality reasoning

- Unrestricted blocking
- Sound restricted blocking

## Unrestricted blocking mechanism [ISWC07]

### Add the following

$$
\begin{align*}
\text{(ub) rule:} & & s \approx t & s \not\approx t (s \neq t) \\
\text{Ordered rewriting:} & & s \approx t \text{ is a trigger for rewriting } s \rightarrow t, \text{ if } s > t \\
\text{Termination condition} & & \text{Apply (ub) rule eagerly from some point onwards for all pairs of terms}
\end{align*}
$$
Soundness, completeness and termination

Sound equality ancestor blocking

Example: Unrestricted blocking

\[(\text{ub}) \quad s \approx t \mid s \not\approx t \quad (s \neq t)\]

Example: \(\Box \neg \Box p\) is S4-satisfiable

\[R(a, a)\]
\[\neg \Box p\]
\[\neg \Box p\]
\[a \approx b\]

\[b = f(p, a)\]
\[\neg p\]

Assume \(a \approx b\)
Rewrite \(b \rightarrow a\)

Restrict application of (\text{ub=})

\[(\text{ub=}) \quad s \approx t \mid s \not\approx t \quad (s \text{ is an ancestor of } t, \ L(s) = L(t), \ s \neq t)\]

where \(L(s) = \{\phi \mid s : \phi \text{ in current branch } B\}\)

Example from before: \(\Box \neg \Box p\) is S4-satisfiable

\[R(a, a)\]
\[\neg \Box p\]
\[\neg \Box p\]
\[a \approx b\]

\[b = f(p, a)\]
\[\neg p\]
\[a \approx b\]
Sound equality ancestor blocking

Restrict application of (ub)

\[(ub-) \quad s \approx t \mid s \not\approx t \quad (s \text{ is an ancestor of } t, \ L(s) = L(t), \ s \not\approx t)\]

where \(L(s) = \{\phi \mid s : \phi \text{ in current branch } B\}\)

Example from before: \(\Box \neg \neg p\) is S4-satisfiable

\[c = f(p, b) = f(p, f(p, a))\]
\[L(b) = L(c) \sim \text{Assume } b \approx c\]
\[\text{Rewrite } c \rightarrow b\]

Sound equality ancestor blocking (cont’d)

- Properties of sound equality ancestor blocking
  - Produces larger models
  - Creates fewer decision points
  - Emulates standard equality ancestor blocking
  - Sound and logic-independent

- Properties of standard equality ancestor blocking
  - Not generally sound
  - Soundness can be ensured for certain MLs/DLs with a tree-model property via a certain rule application strategy + (\(\Diamond\Box\))-rule
  - Gives strong termination results for many MLs/DLs
Turning calculi into deterministic procedures

Tableau calculi provide non-deterministic procedures
At any point there is complete flexibility in choosing:
- which branch to select and expand next
- which rule to apply next
- which formula to select

How to turn the developed tableau calculi into deterministic procedures?
Without losing soundness, completeness and termination?
That a calculus is sound, complete and terminating does not automatically imply that its implementation is sound, complete and terminating.

Techniques and strategies are needed

Ensuring soundness

Soundness: If $N$ is satisfiable, then in any non-deterministically constructed tableau derivation $Tab(N)$ is open.

Problem: Does this imply that every deterministic procedure starting constructs an open tableau for $N$?

- Problematic are techniques not generally sound and logic-dependent
  E.g., standard blocking techniques for MLs/DLs

Solution: Apply the rules in a certain order –
- first Boolean rules, then $(\Diamond\Box)$-rule

Ensuring completeness

Completeness: If $N$ is unsatisfiable, then any tableau derivation $Tab(N)$ constructed non-deterministically for it is closed

Problem: Does this imply that every deterministic procedure starting from $N$ constructs a closed tableau?

- Problematic are rules of universal quantifier extent

\[
\begin{align*}
\gamma : \Box \phi, \, R(s, t) & \quad \Rightarrow \quad t : \phi \\
\gamma : \Box \phi & \quad \text{Applicable to the same } s : \Box \phi \text{ on a branch for each } R(s, t) \text{ occurring on that branch}
\end{align*}
\]

- Fairness ensures completeness for deterministic tableau procedure

A tableau procedure is fair if: When a rule is applicable to a formula then the rule is eventually applied to this formula on every branch on which it occurs (unless the branch is closed and an open, fully expanded branch has already been found)

Solution: Give $\gamma$-rules and $\gamma$-formulae equal priority
Ensuring completeness

**Completeness:** If $N$ is unsatisfiable, then any tableau derivation $Tab(N)$ constructed non-deterministically for it is closed.

**Problem:** Does this imply that every deterministic procedure starting from $N$ constructs a closed tableau?

**Problematic are rules of universal quantifier extent**

\[
\frac{s : \square \phi, R(s, t)}{t : \phi}
\]

Applicable to the same $s : \square \phi$ on a branch for each $R(s, t)$ occurring on that branch.

**Fairness ensures completeness for deterministic tableau procedure**

A tableau procedure is fair if: When a rule is applicable to a formula then the rule is eventually applied to this formula on every branch on which it occurs (unless the branch is closed and an open, fully expanded branch has already been found).

**Solution:** Give $\gamma$-rules and $\gamma$-formulae equal priority.

Ensuring termination

**Strong termination:** If $N$ is a finite, satisf. set, then every fully expanded or closed branch of any non-deterministic constructed tableau $Tab(N)$ is finite.

**Problem:** Does this imply that every deterministic procedure starting from $N$ constructs a finite open tableau?

**Yes, for every fair derivation**

**Any fair tableau procedure provides a decision procedure; this means depth-first search strategies and arbitrary branch selection strategies may be used.**

**Weak termination:** If $N$ is a finite, satisf. set, then any non-deterministic constructed tableau $Tab(N)$ has a finite open, fully expanded branch.

**Problem:** Does this imply that every deterministic procedure starting from $N$ constructs a finite open, fully expanded branch?

**Problematic are sound blocking rules**

\[
\frac{s \approx t \mid s \not\approx t}{s \neq t}
\]

Always choosing the right branch at (ub) branching points is like not using blocking at all – if depth-first search is used.

Always choosing the left branch is also not a solution, if depth-first search is used – counterexample for $FO^2$ due to Reker (2011).

**Fairness of branch selection ensures termination for deterministic proc.**

**Solution:** Use depth-first iterative deepening or depth-first up-to-maximal-bound or breadth-first search.
Ensuring termination

**Weak termination:** If \( N \) is a finite, satisf. set, then any non-deterministic constructed \( \text{Tab}(N) \) has a finite open, fully expanded branch.

**Problem:** Does this imply that every deterministic procedure starting from \( N \) constructs a finite open, fully expanded branch?

- **Problematic are sound blocking rules**
  
  \[
  \frac{s \approx t}{s \neq t} (s \neq t)
  \]

  Always choosing the right branch at (ub) branching points is like not using blocking at all – if depth-first search is used.

  Always choosing the left branch is also not a solution, if depth-first search is used – counterexample for \( FO^2 \) due to Reker (2011).

- **Fairness of branch selection ensures termination for deterministic proc.**

  **Solution:** Use depth-first iterative deepening or depth-first up-to-maximal-bound or breadth-first search.

Concluding remarks and outlook

The possibilities in designing tableau calculi/provers are endless.

**Focus in this talk**

- Synthesis
- Refinement
- Blocking
- Determinisation & search strategies

**Much remains to be done**
Prover generation: http://www.mettel-prover.org/

Some references

- **Tableau synthesis**

- **Applications**

- **Unrestricted blocking**

Thanks

- Dmitry Tishkovsky
- Mohammad Khodadadi
- Hilverd Reker
- Fabio Papacchini
- Users: Michal Zawidzki, Stefan Minica, Clare Dixon, Boris Konev, …